



Gowin RiscV AE350 SOC NN Software Programming Manual

MUG1181-1.1E, 03/21/2025

Copyright © 2025 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any denotes, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
02/02/2024	1.0E	Initial version published.
03/21/2025	1.1E	The TinyYOLO v3 example reference design supported.

Contents

Contents	i
List of Figures	vii
List of Tables	viii
1 About This Guide	1
1.1 Purpose	1
1.2 Terminology and Abbreviation	1
1.3 Support and Feedback	2
2 Overview	3
2.1 Linking Options for the Newlib and MCULib Toolchains	5
2.2 Linking Options for the Glibc Toolchains	5
3 Function Interface Descriptions	7
3.1 Activation Functions	7
3.1.1 riscv_nn_activate_s8	7
3.1.2 riscv_nn_activate_s16	8
3.1.3 riscv_nn_leaky_relu_s8	8
3.1.4 riscv_nn_relu_any_s8	9
3.1.5 riscv_nn_relu_s8	9
3.1.6 riscv_nn_relu_s16	10
3.1.7 riscv_nn_sigmoid_f16	10
3.1.8 riscv_nn_tanh_f16	10
3.2 Basic Functions	11
3.2.1 riscv_nn_add_s8_sym	11
3.2.2 riscv_nn_add_s8_sym_round	13
3.2.3 riscv_nn_ew_add_s8_asym	14
3.2.4 riscv_nn_ew_mul_s8_asym	16

3.3 Concatenation Functions	18
3.3.1 riscv_nn_concat_s8_w	18
3.3.2 riscv_nn_concat_s8_x	18
3.3.3 riscv_nn_concat_s8_y	19
3.3.4 riscv_nn_concat_s8_z	20
3.4. Convolution Functions	21
3.4.1 riscv_nn_conv_1x1_HWC_s8_s8_s8_sft_bias_fast_any.....	22
3.4.2 riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias	24
3.4.3 riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias_fast	26
3.4.4 riscv_nn_conv_HWC_s8_s8_s8_sft_bias	27
3.4.5 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_any	29
3.4.6 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast	31
3.4.7 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast_any	32
3.4.8 riscv_nn_conv_HWC_s16_s16_s16_sft_bias	34
3.4.9 riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast	36
3.4.10 riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast_any	37
3.4.11 riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias.....	39
3.4.12 riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias_any.....	41
3.4.13 riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any.....	43
3.4.14 riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_bias_fast_any.....	44
3.4.15 riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_bias_fast_any	46
3.4.16 riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_bias_fast_any	48
3.4.17 riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_bias_fast_any	49
3.4.18 riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_fast_any	51
3.4.19 riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_fast_any	52
3.4.20 riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_fast_any	54
3.4.21 riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_fast_any	55
3.4.22 riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_fast_any	57
3.4.23 riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast	58
3.4.24 riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_bias_fast	59
3.4.25 riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_bias_fast	60
3.4.26 riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_bias_fast	61

3.4.27 riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_bias_fast	62
3.4.28 riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_fast	63
3.4.29 riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_fast	64
3.4.30 riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_fast	65
3.4.31 riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_fast	66
3.4.32 riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_fast	67
3.4.33 riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast	68
3.4.34 riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast	69
3.4.35 riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast	70
3.4.36 riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast	71
3.4.37 riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast	72
3.4.38 riscv_nn_conv_HWC_s8_s8_s8_sym_fast	73
3.4.39 riscv_nn_conv_HWC_s8_s16_s8_sym_fast	74
3.4.40 riscv_nn_conv_HWC_u8_u8_s8_sym_fast	75
3.4.41 riscv_nn_conv_HWC_u8_s8_s8_sym_fast	76
3.4.42 riscv_nn_conv_HWC_u8_s16_s8_sym_fast	77
3.4.43 riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast_any	78
3.4.44 riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast_any	79
3.4.45 riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast_any	80
3.4.46 riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast_any	82
3.4.47 riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast_any	83
3.4.48 riscv_nn_conv_HWC_s8_s8_s8_sym_fast_any	84
3.4.49 riscv_nn_conv_HWC_s8_s16_s8_sym_fast_any	85
3.4.50 riscv_nn_conv_HWC_u8_u8_s8_sym_fast_any	87
3.4.51 riscv_nn_conv_HWC_u8_s8_s8_sym_fast_any	88
3.4.52 riscv_nn_conv_HWC_u8_s16_s8_sym_fast_any	89
3.4.53 riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias	90
3.4.54 riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias	91
3.4.55 riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias	92
3.4.56 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias	93
3.4.57 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias	94
3.4.58 riscv_nn_conv_dw_HWC_s8_s8_s8_sym	95

3.4.59 riscv_nn_conv_dw_HWC_s8_s16_s8_sym	96
3.4.60 riscv_nn_conv_dw_HWC_u8_u8_s8_sym	97
3.4.61 riscv_nn_conv_dw_HWC_u8_s8_s8_sym	98
3.4.62 riscv_nn_conv_dw_HWC_u8_s16_s8_sym	99
3.4.63 riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias_any	100
3.4.64 riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias_any	102
3.4.65 riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias_any	103
3.4.66 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias_any	104
3.4.67 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias_any	105
3.4.68 riscv_nn_conv_dw_HWC_s8_s8_s8_sym_any	107
3.4.69 riscv_nn_conv_dw_HWC_s8_s16_s8_sym_any	108
3.4.70 riscv_nn_conv_dw_HWC_u8_u8_s8_sym_any	109
3.4.71 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_any	110
3.4.72 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_any	111
3.4.73 riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any	113
3.4.74 riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size	114
3.4.75 riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any	115
3.4.76 riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any_get_buffer_size	116
3.4.77 riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any	116
3.4.78 riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any_get_buffer_size	118
3.4.79 riscv_nn_conv_dw_HWC_3x3_s8_s8_s8_asym_bias_any	118
3.4.80 riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_any	120
3.4.81 riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any	121
3.4.82 riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size	123
3.4.83 riscv_nn_conv_dw_HWC_u8_u8_u8_asym_bias_any	123
3.4.84 riscv_nn_conv_1x1_HWC_f16_f16_f16_bias_any	125
3.4.85 riscv_nn_conv_HWC_f16_f16_f16_bias	126
3.4.86 riscv_nn_conv_dw_HWC_f16_f16_f16_bias	127
3.5 Fully Connected Functions	128
3.5.1 riscv_nn_fc_s8_s8_s8_sft_bias	128
3.5.2 riscv_nn_fc_s8_s8_s8_sft_bias_fast	129
3.5.3 riscv_nn_fc_s16_s16_s16_sft_bias	130

3.5.4 riscv_nn_fc_s16_s16_s16_sft_bias_fast.....	130
3.5.5 riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias.....	131
3.5.6 riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias_fast.....	132
3.5.7 riscv_nn_fc_s8_s8_s8_sym_bias.....	132
3.5.8 riscv_nn_fc_s8_s16_s8_sym_bias.....	133
3.5.9 riscv_nn_fc_u8_u8_s8_sym_bias.....	134
3.5.10 riscv_nn_fc_u8_s8_s8_sym_bias.....	135
3.5.11 riscv_nn_fc_u8_s16_s8_sym_bias.....	136
3.5.12 riscv_nn_fc_s8_s8_s8_sym.....	136
3.5.13 riscv_nn_fc_s8_s16_s8_sym.....	137
3.5.14 riscv_nn_fc_u8_u8_s8_sym.....	138
3.5.15 riscv_nn_fc_u8_s8_s8_sym.....	139
3.5.16 riscv_nn_fc_u8_s16_s8_sym.....	139
3.5.17 riscv_nn_fc_s8_s8_s8_sym_bias_fast.....	140
3.5.18 riscv_nn_fc_s8_s16_s8_sym_bias_fast.....	141
3.5.19 riscv_nn_fc_u8_u8_s8_sym_bias_fast.....	142
3.5.20 riscv_nn_fc_u8_s8_s8_sym_bias_fast.....	143
3.5.21 riscv_nn_fc_u8_s16_s8_sym_bias_fast.....	143
3.5.22 riscv_nn_fc_s8_s8_s8_sym_fast.....	144
3.5.23 riscv_nn_fc_s8_s16_s8_sym_fast.....	145
3.5.24 riscv_nn_fc_u8_u8_s8_sym_fast.....	146
3.5.25 riscv_nn_fc_u8_s8_s8_sym_fast.....	147
3.5.26 riscv_nn_fc_u8_s16_s8_sym_fast.....	147
3.5.27 riscv_nn_fc_s8_wt_converter.....	148
3.5.28 riscv_nn_fc_s16_wt_converter.....	149
3.5.29 riscv_nn_fc_mat_vec_s8_wt_converter.....	149
3.5.30 riscv_nn_fc_s8_s8_s8_asym_bias.....	150
3.5.31 riscv_nn_fc_s8_s8_s8_asym_bias_get_buffer_size.....	151
3.5.32 riscv_nn_fc_f16_f16_f16_bias.....	151
3.6 Pooling Functions.....	151
3.6.1 riscv_nn_avepool_HWC_s8.....	152
3.6.2 riscv_nn_avepool_HWC_s8_any.....	153

3.6.3 riscv_nn_avepool_HWC_s8_any_act.....	154
3.6.4 riscv_nn_avepool_HWC_s8_any_act_get_buffer_size.....	155
3.6.5 riscv_nn_maxpool_HWC_s8.....	155
3.6.6 riscv_nn_maxpool_HWC_s8_any_act.....	156
3.7 Softmax Functions.....	157
3.7.1 riscv_nn_softmax_s8_fast.....	157
3.7.2 riscv_nn_softmax_s16_fast.....	158
3.7.3 riscv_nn_softmax_s8_hp.....	158
3.7.4 riscv_nn_softmax_u8_hp.....	159
3.7.5 riscv_nn_softmax_f16.....	160
3.8 Utils Functions.....	160
3.8.1 riscv_nn_exp_f16.....	160
3.8.2 riscv_nn_reshape_s8.....	161
3.8.3 riscv_nn_top_k_s8.....	161
3.8.4 riscv_nn_top_k_f16.....	162
4 Application Program.....	163
4.1 NN CIFAR-10.....	163
4.1.1 Program Description.....	163
4.1.2 Application Program.....	163
4.1.3 Program Running.....	163
4.2 NN MobileNet-V1 INT8.....	165
4.2.1 Program Description.....	165
4.2.2 Application Program.....	165
4.2.3 Program Running.....	165
4.3 NN TinyYOLO-V3.....	168
4.3.1 Program Description.....	168
4.3.2 Application Program.....	168
4.3.3 Program Running.....	168
4.4 NN TinyYOLO-V1.....	170
4.4.1 Program Description.....	170
4.4.2 Application Program.....	171
4.4.3 Program Running.....	171

List of Figures

Figure 3-1 riscv_nn_add_s8_sym Algorithm Flow	12
Figure 3-2 riscv_nn_add_s8_sym_round Algorithm Flow	14
Figure 3-3 riscv_nn_ew_add_s8_asym_ Algorithm Flow	15
Figure 3-4 riscv_nn_ew_mul_s8_asym Algorithm Flow	17
Figure 3-5 Algorithmic Process for Convolution Functions Based on Shift-based Quantization	21
Figure 3-6 Algorithmic Process for Convolution Functions with Symmetric Quantization	21
Figure 3-7 Algorithmic Process for Convolution Functions with Asymmetric Quantization	22

List of Tables

Table 1-1 Terminology and Abbreviations	1
Table 2-1 Linking Options	6

1 About This Guide

1.1 Purpose

This manual describes the function specification of Gowin AE350 SOC NN software programming function library, and how to use the NN function interface software programming, the application program demonstrations, etc.

1.2 Terminology and Abbreviation

Table 1-1 shows the abbreviations and terminology used in this manual.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
DSP	Digital Signal Processor
FPU	Floating Point Unit
ISA	Instruction Set Architecture
NN	Neural Network
ReLU	Rectified Linear Unit
RISC- V	Reduced Instruction Set Computer V
SOC	System on Chip
Zfh	Half Precision Floating Point

1.3 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly using the information provided below.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

Gowin RiscV AE350 SOC NN software programming function library contains functional description on the neural network function for use with the RDS software toolchain, as well as description on how to use these function interfaces.

The NN software programming library provides comprehensive functions implemented in pure C language or optimized for DSP ISA extensions. These functions include activation, convolution, fully-connected, pooling and miscellaneous auxiliary functions. Designed and developed using the RDS software toolchain, the NN software programming library encapsulates the complexity of computations in neural network computation and provides an easy-to-use application function interface, which saves development time and resources, allowing users to focus more on system design.

The functions in the NN software programming library includes the following categories: activation function, basic math function, concatenation function, convolution function, fully-connected functions, pooling functions, softmax function and tool auxiliary function, and they will be introduced in details in chapter 3.

When the NN function interface is called, only the header files of the NN software programming function library are included. These header files define the prototypes of function interfaces and instance structures, named by functional category, as shown below:

- [riscv_nn_activation.h](#)
- [riscv_nn_basic.h](#)
- [riscv_nn_concatenation.h](#)
- [riscv_nn_convolution.h](#)
- [riscv_nn_fully_connected.h](#)
- [riscv_nn_pooling.h](#)

- [riscv_nn_softmax.h](#)
- [riscv_nn_util.h](#)

For example, if you need to use activation functions to activate neural network data, you can include the header file [riscv_nn_activation.h](#) in the C file.

The NN software programming library defines several data types to store data for subsequent processing. These data types are defined in [riscv_math_types.h](#) as follows: [q7_t](#) for signed 8-bit integer, [u8_t](#) for unsigned 8-bit integer, [q15_t](#) for signed 16-bit integer, [u16_t](#) for unsigned 16-bit integer, [q31_t](#) for signed 32-bit integer, and [u32_t](#) for unsigned 32-bit integer [q63_t](#) for signed 64-bit integer, [u64_t](#) for unsigned 64-bit integer, [float16_t](#) for 16-bit half-precision floating-point, [float32_t](#) for 32-bit single-precision floating-point and [float64_t](#) for 64-bit double-precision floating-point respectively:

- `typedef int8_t q7_t; /* for signed 8-bit integer */`
- `typedef uint8_t u8_t; /* for unsigned 8-bit integer */`
- `typedef int16_t q15_t; /* for signed 16-bit integer */`
- `typedef uint16_t u16_t; /* for unsigned 16-bit integer */`
- `typedef int32_t q31_t; /* for signed 32-bit integer */`
- `typedef uint32_t u32_t; /* for unsigned 32-bit integer */`
- `typedef int64_t q63_t; /* for signed 64-bit integer */`
- `typedef uint64_t u64_t; /* for unsigned 64-bit integer */`
- `typedef _Float16 float16_t; /* for half-precision (16-bit) floating-point */`
- `typedef float float32_t; /* for single-precision (32-bit) floating-point */`
- `typedef double float64_t; /* for double-precision (64-bit) floating-point */`

In addition, the NN software programming library defines the type `riscv_nn_activation_fun`, defined in `riscv_nn_activation.h`, which is used to select the activation function:

```
typedef enum
{
    NN_SIGMOID = 0, /* Use sigmoid activation function */
    NN_TANH = 1, /* Use tanh activation function */
} riscv_nn_activation_fun;
```

2.1 Linking Options for the Newlib and MCULib Toolchains

In addition to the header files described above, different toolchains require different linking options, and this section details the options used for Newlib and MCULib, and the options include followings:

- `-lnn` link NN software programming function library (libnn.a)
- `-mext-dsp` enable DSP ISA extension
- `-mzfh` enable the half-precision floating-point data type and Zfh extension

If the target system does not support the DSP ISA extension to accelerate the NN software programming function library, use `-lnn` to build an application that links to the NN software programming function library (libnn.a). If the target system supports DSP ISA extensions, use the `-lnn`, `mext-dsp` options to enable the extension and link the NN software programming library libnn.a for better performance. It should be noted that applications built with the `-lnn`, `-mext-dsp` options should not be run on systems that do not support DSP ISA extension, otherwise problems will occur when loading.

If your application requires the use of the half-precision floating-point data type (i.e., functions named with "f16"), it is crucial to employ a toolchain that supports the Floating Point Unit (FPU), specifically the v5f or v5d toolchain. Additionally, ensure that you utilize the option `-mzfh` to enable the half-precision floating-point data type and the Zfh extension.

2.2 Linking Options for the Glibc Toolchains

The linking options available for the glibc toolchain include:

- `-static` use static linking
- `-lnn` link NN software programming function library implemented by pure C language
- `-lnn_p` link NN software programming function library with DSP ISA Extension

Static and dynamic links require different options, Table 2-1 summarizes the linking options for different link types and whether DSP ISA extension support is required.

Table 2-1 Linking Options

Type	DSP ISA Extension	Option
Static Linking	Without	<code>-static -lnn</code>
	With	<code>-static -lnn_p</code>
Dynamic Linking (default)	Without	<code>-lnn</code>
	With	<code>-lnn_p</code>

The required options have DSP ISA extensions (e.g., `-static -lnn_p` and `lnn_p`) for building applications for target systems that support the extensions, while the other options (e.g., `-static -lnn` and `-lnn`), are used only for target systems that do not support the extension. Users need to ensure that the options specified correspond to the target system used.

3 Function Interface Descriptions

The following sections, categorized by functions, describe the function interfaces of the NN software programming library in details.

3.1 Activation Functions

The activation functions are used to filter out some input data. They include sigmoid, tanh and ReLU (Rectified Linear Unit) functions.

3.1.1 riscv_nn_activate_s8

Prototype:

```
void riscv_nn_activate_s8 (q7_t *in_out, uint32_t size, uint16_t  
int_bits, riscv_nn_activation_fun act_fun)
```

Description

This function uses the sigmoid or tanh function to perform activation for signed 8-bit integer input vectors.

Parameter:

- [in, out] `*in_out` pointer of the input/output vector
- [in] `size` number of elements in the input/output vector
- [in] `int_bits` number of the bits in the integer part, which is supposed to be smaller than 4
- [in] `act_fun` selection of activation functions. `NN_SIGMOID` is used to select the sigmoid activation function, and `NN_TANH` is used to select the tanh activation function.

Return Value:

None

3.1.2 riscv_nn_activate_s16

Prototype:

```
void riscv_nn_activate_s16 (q15_t *in_out, uint32_t size, uint16_t  
int_bits, riscv_nn_activation_fun act_fun)
```

Description

This function uses the sigmoid or tanh function to perform activation for signed 16-bit integer input vectors.

Parameter:

- [in, out] `*in_out` pointer of the input/output vector
- [in] `size` number of elements in the input/output vector
- [in] `int_bits` number of the bits in the integer part, which is supposed to be smaller than 4
- [in] `act_fun` selection of activation functions. NN_SIGMOID is used to select the sigmoid activation function, and NN_TANH is used to select the tanh activation function.

Return Value:

None

3.1.3 riscv_nn_leaky_relu_s8

Prototype:

```
void riscv_nn_leaky_relu_s8 (q7_t *in_out, uint32_t size, q15_t slope)
```

Description

This function uses the leaky ReLU function to perform activation for signed 8-bit integer input vectors.

Parameter:

- [in, out] `*in_out` pointer of the input/output vector
- [in] `size` number of elements in the input/output vector
- [in] `slope` slope value to be multiplied with the negative inputs, and the result will be right shifted 15 bits to scale back to a signed 8-bit integer

Return Value:

None

Example:

```
#define SIZE 1024  
q15_t slope = 16384;
```

```
q7_t in_out[SIZE] = {...};  
riscv_nn_leaky_relu_s8 (in_out, SIZE, slope);
```

3.1.4 riscv_nn_relu_any_s8

Prototype:

```
void riscv_nn_relu_any_s8 (q7_t *data, uint16_t size, q7_t max_val)
```

Description

This function uses the ReLU function to perform activation for signed 8-bit integer input vectors.

Parameter:

- [in, out] *data pointer of the input/output vector
- [in] size number of elements in the input/output vector
- [in] max_val maximum value to limit the output vector

Return Value:

None

3.1.5 riscv_nn_relu_s8

Prototype:

```
void riscv_nn_relu_s8 (q7_t *in_out, uint32_t size)
```

Description

This function uses the ReLU function to perform activation for signed 8-bit integer input vectors.

Parameter:

- [in, out] *in_out pointer of the input/output vector
- [in] size number of elements in the input/output vector

Return Value:

None

Example:

```
#define H 16  
#define W 16  
#define CH 5  
#define NUM (H * W * CH)  
q7_t in_out[NUM] = {...};
```

```
riscv_nn_relu_s8 (in_out, NUM);
```

3.1.6 riscv_nn_relu_s16

Prototype:

```
void riscv_nn_relu_s16 (q15_t *in_out, uint32_t size)
```

Description

This function uses the ReLU function to perform activation for signed 16-bit integer input vectors.

Parameter:

- `[in, out] *in_out` pointer of the input/output vector
- `[in] size` number of elements in the input/output vector

Return Value:

None

3.1.7 riscv_nn_sigmoid_f16

Prototype:

```
int32_t riscv_nn_sigmoid_f16 (const float16_t * in_vec, uint32_t size,  
float16_t * out_vec)
```

Description

This function uses the sigmoid function to perform activation for half-precision floating-point input vectors.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] size` number of elements in the input/output vector
- `[out] *out_vec` pointer of the output vector

Return Value:

0

Note!

The input vectors are restricted to the range [-10, 10].

3.1.8 riscv_nn_tanh_f16

Prototype:

```
int32_t riscv_nn_tanh_f16 (const float16_t * in_vec, uint32_t size,  
float16_t * out_vec)
```

Description

This function uses the tanh function to perform activation for half-precision floating-point input vectors.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] size` number of elements in the input/output vector
- `[out] *out_vec` pointer of the output vector

Return Value:

0

Note!

The inputs are restricted to the range [-10, 10].

3.2 Basic Functions

The basic functions are used to perform element-wise basic arithmetic operations.

3.2.1 riscv_nn_add_s8_sym

Prototype:

```
void riscv_nn_add_s8_sym (const q7_t *in_tensor1, const q7_t *
in_tensor2, const int16_t *scale1, const int16_t *scale2, const uint32_t
size, const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q7_t * out)
```

Description

This function performs element-wise addition for signed 8-bit integer input vectors with symmetric quantization on the outputs.

Parameter:

- `[in] * in_tensor1` pointer of the first input vector
- `[in] * in_tensor2` pointer of the second input vector
- `[in] * scale1` pointer of the first scaling vector
- `[in] * scale2` pointer of the the second vector
- `[in] size` number of elements in the input vector
- `[in] pre_rshift` right shift amount for the accumulator before the scaling
- `[in] out_scale` scaling value for the accumulator
- `[in] post_rshift` right shift amount for the accumulator after the scaling

- `[out] * out` pointer of the element-wise addition results

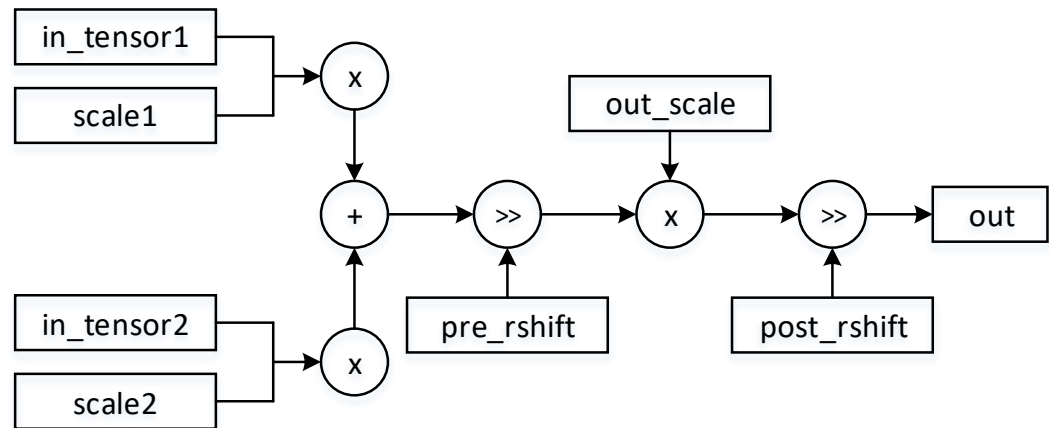
Return Value:

None

Note!

The calculation of each element can be represented as Figure 3-1, where rectangles represent inputs or outputs, circles represent arithmetical operations, solid lines are for required steps and dashed lines, if any, are for optional ones.

Figure 3-1 riscv_nn_add_s8_sym Algorithm Flow



Example:

```
#define SIZE 1024
```

```
uint16_t pre_rshift = 8; // The addition results of both scaled input
```

```
// tensors are in the range of 24-bit; thus, the
```

```
// pre_rshift should be in the range of [0, 24].
```

```
// Here we scale down the results into 16-bit
```

```
// range.
```

```
uint16_t out_scale = 3; // Scale up the result into 18-bit range.
```

```
uint16_t post_rshift = 11; // Scale down the result into 7-bit range.
```

```
q7_t in_tensor1[SIZE] = {...};
```

```
q7_t in_tensor2[SIZE] = {...};
```

```
q15_t scale1[SIZE] = {...};
```

```
q15_t scale2[SIZE] = {...};
```

```
q7_t out[SIZE];
```

```
riscv_nn_add_s8_sym (in_tensor1, in_tensor2, scale1, scale2, SIZE,  
pre_rshift, out_scale, post_rshift, out);
```

3.2.2 riscv_nn_add_s8_sym_round

Prototype:

```
void riscv_nn_add_s8_sym_round (const q7_t *in_tensor1, const q7_t  
* in_tensor2, const uint32_t scale1, const uint32_t scale2, const uint32_t  
size, const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t  
post_rshift, q7_t * out)
```

Description

This function performs element-wise addition for signed 8-bit integer input vectors with symmetric quantization and rounding on the outputs.

Parameter:

- [in] * `in_tensor1` pointer of the first input vector
- [in] * `in_tensor2` pointer of the second input vector
- [in] `scale1` scaling value for the first input vector. It should be in the range of 0 to 2^{23}
- [in] `scale2` scaling value for second input vector. It should be in the range of 0 to 2^{23}
- [in] `size` number of elements in the input vector
- [in] `pre_rshift` right shift amount for the accumulator before the scaling
- [in] `out_scale` scaling value for the accumulator
- [in] `post_rshift` right shift amount for the accumulator after the scaling
- [out] * `out` pointer of the element-wise addition results

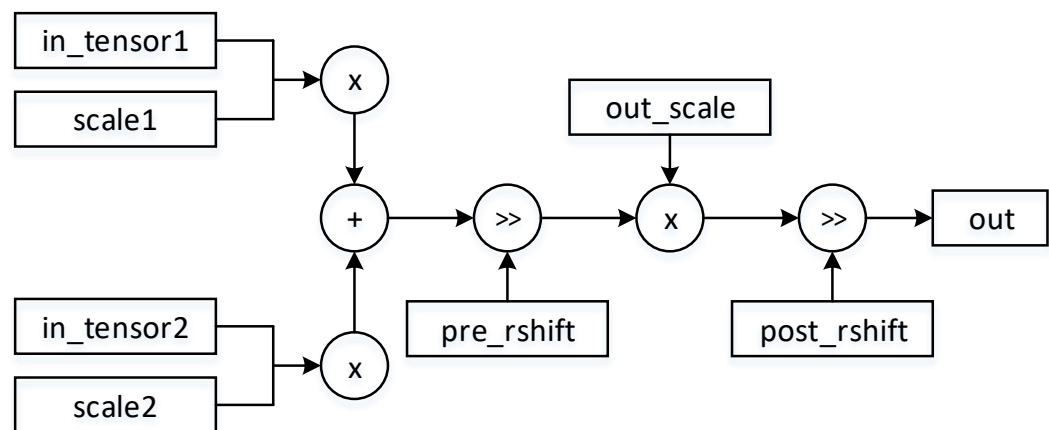
Return Value:

None

Note!

- The calculation of each element could be represented as Figure 3-2.
- The right shift operations for this function include rounding.

Figure 3-2 riscv_nn_add_s8_sym_round Algorithm Flow



3.2.3 riscv_nn_ew_add_s8_asym

Prototype:

```
int riscv_nn_ew_add_s8_asym (const int8_t *in_tensor1, const int8_t
* in_tensor2, const int32_t in_offset1, const int32_t in_scale1, const
int32_t in_rshift1, const int32_t in_offset2, const int32_t in_scale2, const
int32_t in_rshift2, const int32_t lshift, int8_t * out, const int32_t out_offset,
const int32_t out_scale, const int32_t out_rshift, const int32_t act_min,
const int32_t act_max, const uint32_t size)
```

Description

This function performs element-wise addition for signed 8-bit integer input vectors.

Parameter:

- [in] * in_tensor1 pointer of the first input vector
- [in] * in_tensor2 pointer of the second input vector
- [in] in_offset1 offset value for first input vector. It should be in the range of -127 to 128
- [in] scale1 scaling value for first input vector
- [in] in_rshift1 right shift amount for the first input vector
- [in] in_offset2 offset value for the second input vector. It should be in the range of -127 to 128
- [in] scale2 scaling value for the second input vector
- [in] in_rshift2 right shift amount for the second input vector
- [in] in_rshift1 left shift amount for the first and second input vectors
- [out] *out pointer of the element-wise addition results
- [in] out_offset offset value for the output vector

- [in] `out_scale` scaling value for the output vector
- [in] `out_rshift` right shift amount for the output vector
- [in] `act_min` minimum value that the output vector is limited to
- [in] `act_max` maximum value that the output vector is limited to
- [in] `size` number of elements in the input vector

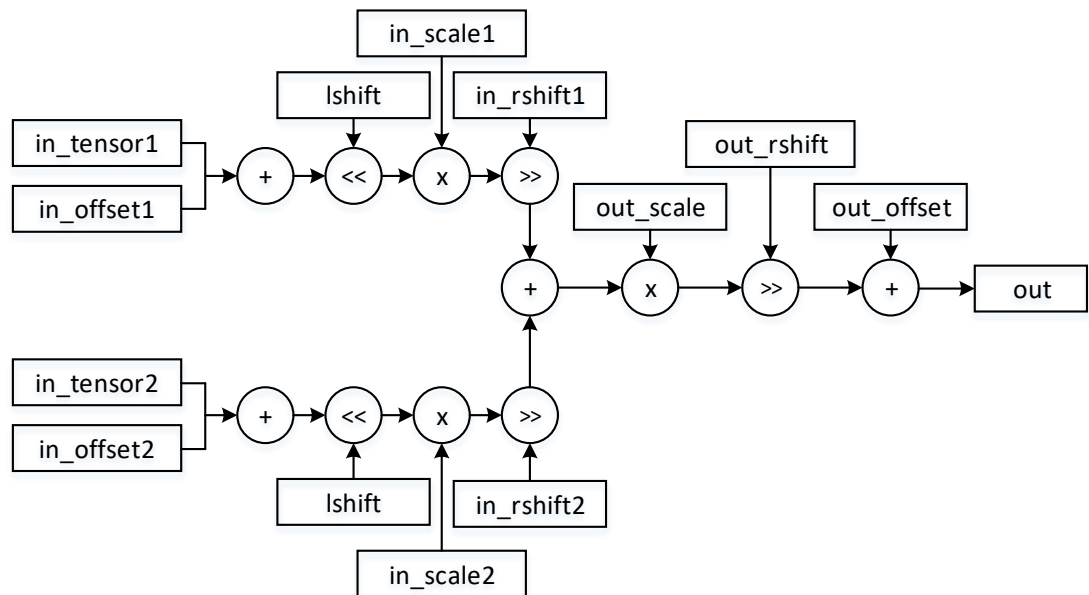
Return Value:

0

Note!

- The calculation of each element could be represented as Figure 3-3.
- The multiplication for `in_scale1`, `in_scale2` and `out_scale` could be roughly expressed as: $32b = ((int64\_t)32b * 32b) \gg 31$.

Figure 3-3 riscv_nn_ew_add_s8_asym_ Algorithm Flow



Example:

```

#define SIZE 1024
int32_t in_offset1 = 16; // Offset for in_tensor1
int32_t in_scale1 = (1<<28); // Scale down in_tensor1 by 1/23
int32_t in_rshift1 = 3; // Scale down in_tensor1 by 1/23
int32_t in_offset2 = 17; // Offset for in_tensor2
int32_t in_scale2 = (1<<28); // Scale down in_tensor2 by 1/23
int32_t in_rshift2 = 3; // Scale down in_tensor2 by 1/23
int32_t lshift = 10; // Scale up the input tensor by 210 times
  
```

```

int32_t out_offset = 18; // Offset for the output tensor
int32_t out_scale = (1<<30); // Scale down in_tensor2 by 1/2
int32_t out_rshift = 4; // Scale down in_tensor2 by 1/2^4
int32_t act_min = 0xfffffa3; // Limit the outputs in the range of
// [0xfffffa3, 0x0000005d]
int32_t act_max = 0x0000005d; // Limit the outputs in the range of
// [0xfffffa3, 0x0000005d]
int8_t in_tensor1[SIZE] = {...};
int8_t in_tensor2[SIZE] = {...};
int8_t out[SIZE];

riscv_nn_ew_add_s8_asym (in_tensor1, in_tensor2, in_offset1,
in_scale1, in_rshift1, in_offset2, in_scale2, in_rshift2, lshift, out, out_offset,
out_scale, out_rshift, act_min, act_max, SIZE);

```

3.2.4 riscv_nn_ew_mul_s8_asym

Prototype:

```

int riscv_nn_ew_mul_s8_asym (const int8_t * in_tensor1, const int8_t
* in_tensor2, const int32_t in_offset1, const int32_t in_offset2, int8_t * out,
const int32_t out_offset, const int32_t out_scale, const int32_t out_shift,
const int32_t act_min, const int32_t act_max, const uint32_t size)

```

Description

This function performs element-wise multiplication for signed 8-bit integer input vectors.

Parameter:

- `[in] * in_tensor1` pointer of the first input vector
- `[in] * in_tensor2` pointer of the second input vector
- `[in] in_offset1` offset value for first input vector. It should be in the range of -127 to 128
- `[in] in_offset2` offset value for the second input vector. It should be in the range of -127 to 128
- `[out] * out` pointer of the element-wise multiplication results
- `[in] out_offset` offset value for the output vector
- `[in] out_scale` scaling value for the output vector
- `[in] out_rshift` shift amount for the output vector
- `[in] act_min` minimum value that the output vector is limited to

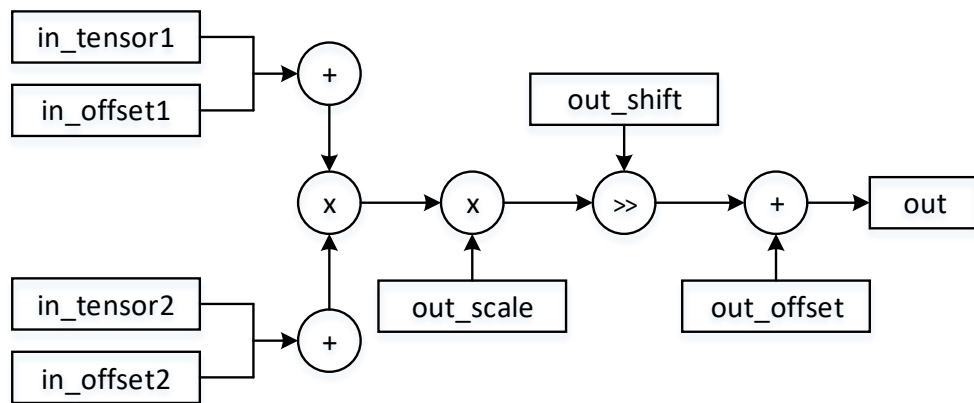
- [in] `act_max` maximum value that the output vector is limited to
- [in] `size` number of elements in the input vector

Return Value:

0

Note!

- The calculation of each element could be represented as Figure 3-4.
- The multiplication for `out_scale` could be roughly expressed as: $32b = ((\text{int64\_t})32b * 32b) \gg 31$.

Figure 3-4 riscv_nn_ew_mul_s8_asym Algorithm Flow**Example:**

```

#define SIZE 1024

int32_t in_offset1 = 16; // Offset for in_tensor1
int32_t in_offset2 = 17; // Offset for in_tensor2
int32_t out_offset = 18; // Offset for the output tensor
int32_t out_scale = (1<<30); // Scale down the output tensor by 1/2
int32_t out_shift = -4; // Scale down the output tensor by 1/2^4
int32_t act_min = 0xfffffa3; // Limit the outputs in the range of
// [0xfffffa3, 0x0000005d]
int32_t act_max = 0x0000005d; // Limit the outputs in the range of
// [0xfffffa3, 0x0000005d]
in_tensor1[SIZE] = {...};
in_tensor2[SIZE] = {...};
out[SIZE];

riscv_nn_ew_mul_s8_asym(in_tensor1, in_tensor2, in_offset1,
in_offset2, out, out_offset, out_scale, out_shift, act_min, act_max, SIZE);
  
```

3.3 Concatenation Functions

The concatenation functions are used to concatenate the tensor along the specified axis.

3.3.1 riscv_nn_concat_s8_w

Prototype:

```
void riscv_nn_concat_s8_w (const int8_t *in_tensor, const uint16_t in_tensor_x, const uint16_t in_tensor_y, const uint16_t in_tensor_z, const uint16_t in_tensor_w, int8_t *out_tensor, const uint32_t out_offset_w)
```

Description

This function concatenates the `int8_t/uint8_t` input tensor along the w-axis with the output tensor.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_x` x dimension of the input tensor
- `[in] in_tensor_y` y dimension of the input tensor
- `[in] in_tensor_z` z dimension of the input tensor
- `[in] in_tensor_w` w dimension of the input tensor
- `[out] *out_tensor` pointer of the output tensor
- `[in] out_offset_w` offset value to be added to the w-axis of the output tensor before the concatenation

Return Value:

None

Note!

The x, y and z dimension of the output tensor will be the same as those of the input tensor.

3.3.2 riscv_nn_concat_s8_x

Prototype:

```
void riscv_nn_concat_s8_x (const int8_t *in_tensor, const uint16_t in_tensor_x, const uint16_t in_tensor_y, const uint16_t in_tensor_z, const uint16_t in_tensor_w, int8_t *out_tensor, const uint16_t out_tensor_x, const uint32_t out_offset_x)
```

Description

This function concatenates the `int8_t/uint8_t` input tensor along the X-

axis with the output tensor.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_x` x dimension of the input tensor
- [in] `in_tensor_y` y dimension of the input tensor
- [in] `in_tensor_z` z dimension of the input tensor
- [in] `in_tensor_w` w dimension of the input tensor
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_x` x dimension of the output tensor
- [in] `out_offset_x` offset value to be added to the X-axis of the output tensor before the concatenation

Return Value:

None

Note!

The y, z, and w dimensions of the output tensor will be the same as those of the input tensor.

3.3.3 riscv_nn_concat_s8_y

Prototype:

```
void riscv_nn_concat_s8_y (const int8_t *in_tensor, const uint16_t  
in_tensor_x, const uint16_t in_tensor_y, const uint16_t in_tensor_z, const  
uint16_t in_tensor_w, int8_t *out_tensor, const uint16_t out_tensor_y,  
const uint32_t out_offset_y)
```

Description

This function concatenates the `int8_t/uint8_t` input tensor along the y-axis with the output tensor.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_x` x dimension of the input tensor
- [in] `in_tensor_y` y dimension of the input tensor
- [in] `in_tensor_z` z dimension of the input tensor
- [in] `in_tensor_w` w dimension of the input tensor
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_y` y dimension of the output tensor

- [in] `out_offset_y` offset value to be added to the y-axis of the output tensor before the concatenation

Return Value:

None

Note!

The x, z, and w dimensions of the output tensor will be the same as those of the input tensor.

3.3.4 `riscv_nn_concat_s8_z`

Prototype:

```
void riscv_nn_concat_s8_z (const int8_t *in_tensor, const uint16_t  
in_tensor_x, const uint16_t in_tensor_y, const uint16_t in_tensor_z, const  
uint16_t in_tensor_w, int8_t * out_tensor, const uint16_t out_tensor_z,  
const uint32_t out_offset_z)
```

Description

This function concatenates the `int8_t/uint8_t` input tensor along the z-axis with the output tensor.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_x` x dimension of the input tensor
- [in] `in_tensor_y` y dimension of the input tensor
- [in] `in_tensor_z` z dimension of the input tensor
- [in] `in_tensor_w` w dimension of the input tensor
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_z` z dimension of the output tensor
- [in] `out_offset_y` offset value to be added to the z-axis of the output tensor before the concatenation

Return Value:

None

Note!

The x, y, and w dimensions of the output tensor will be the same as those of the input tensor.

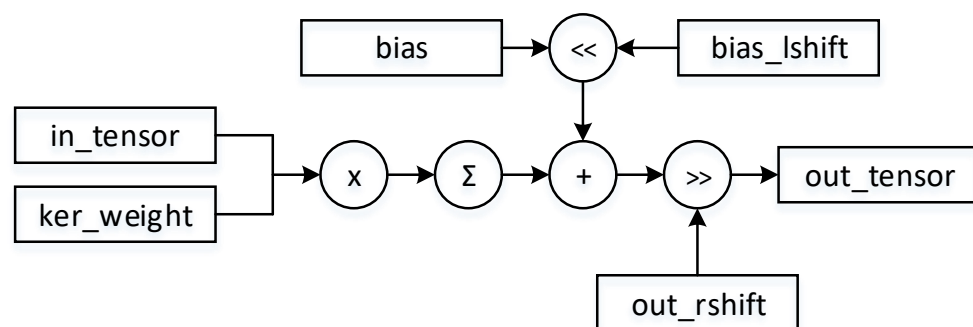
3.4. Convolution Functions

The convolution functions use `im2col` to transform the input matrix into a column vector and then perform matrix-matrix multiplication to obtain the convolution result. There are three types of quantization for the convolution output: shift-based quantization (denoted as "sft"), symmetric quantization (denoted as "sym"), and asymmetric quantization (denoted as "asym").

The following describes the algorithmic processes for each quantization type, where rectangles represent inputs or outputs, circles represent arithmetical operations, solid lines are for required steps and dashed lines, if any, are for optional ones.

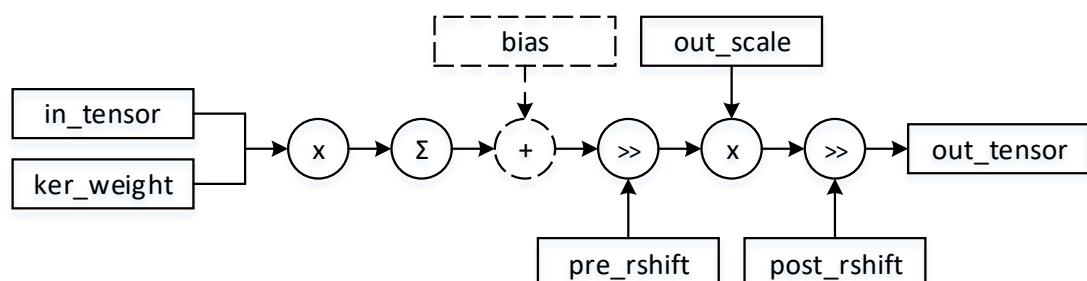
- The algorithmic process for convolution functions based on shift-based quantization is as shown in Figure 3-5.

Figure 3-5 Algorithmic Process for Convolution Functions Based on Shift-based Quantization



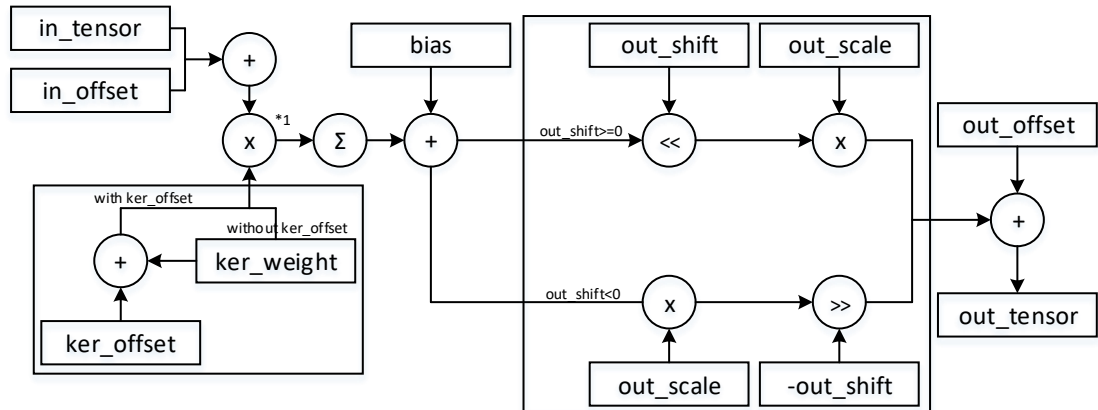
- The algorithmic process for convolution functions with symmetric quantization is as shown in Figure 3-6.

Figure 3-6 Algorithmic Process for Convolution Functions with Symmetric Quantization



- The algorithmic process for convolution functions asymmetric quantization is as shown in Figure 3-7.

Figure 3-7 Algorithmic Process for Convolution Functions with Asymmetric Quantization



Note that in the convolution function with asymmetric quantization, the input to the multiplication operation (*1) can be either "**ker_weight + ker_offset**" or "**ker_weight**," depending on the parameter **ker_offset**. Additionally, the left shift operation is performed only when **out_shift** is positive, and the right shift operation is performed only when **out_shift** is negative. The multiplication operation for **out_scale** can be roughly represented as:

$$32b = ((\text{int64\_t})32b * 32b) >> 31$$

3.4.1 riscv_nn_conv_1x1_HWC_s8_s8_s8_sft_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_s8_s8_s8_sft_bias_fast_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q7_t * bias, const uint16_t
bias_lshift, const uint16_t out_rshift, q7_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf,
q7_t * tmp_buf)
```

Description

This function performs 1x1 kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with shift-based quantization on the outputs.

Parameter:

- **[in] *in_tensor** pointer of the input tensor

- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `post_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0
- `pad_y` 0
- `stride_x` 1

- `stride_y 1`

Example:

`//Convolve a 160x120x20 input tensor with a 1x1 kernel and generate a 160x120x8`

`//output tensor. Let both dimensions' padding be 0 and their stride be 1.`

```
#define IN_X 160
#define IN_Y 120
#define IN_CH 20
#define OUT_CH 8
#define KER_DIM_X 1
#define KER_DIM_Y 1
#define PAD_X 0
#define PAD_Y 0
#define STRIDE_X 1
#define STRIDE_Y 1
#define BIAS_LSHIFT 6 //Scale up the bias by 26
#define OUT_RSHIFT 9 //Scale down the output tensor by 1/29
#define OUT_X 160
#define OUT_Y 120
q7_t in_data[IN_CH * IN_X * IN_Y] = {...};
q7_t weight[IN_CH * KER_DIM_X * KER_DIM_Y * OUT_CH] = {...};
q7_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * IN_CH * KER_DIM_X * KER_DIM_Y] = {0};
q7_t out_data[OUT_CH * OUT_X * OUT_Y];
riscv_nn_conv_1x1_HWC_s8_s8_s8_sft_bias_fast_any (in_data,
IN_X, IN_Y, IN_CH, weight, OUT_CH, KER_DIM_X, KER_DIM_Y, PAD_X,
PAD_Y, STRIDE_X, STRIDE_Y, bias, BIAS_LSHIFT, OUT_RSHIFT,
out_data, OUT_X, OUT_Y, in_tmp_buf, NULL);
```

3.4.2 `riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias`

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
```

```
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const q7_t * bias, const uint16_t bias_lshift, const uint16_t
out_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf, q7_t * tmp_buf)
```

Description

This function performs signed 8-bit integer convolution for RGB images and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `post_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Example:

```
//Convolve a 28x28x3 input tensor with a 5x5 kernel and generate a
24x24x20 output
```

```
//tensor. Let both dimensions' padding be 0 and their stride be 1.
```

```
#define IN_DIM 28
```

```
#define KER_DIM 5
```

```
#define PAD 0
```

```
#define STRIDE 1
```

```
#define BIAS_LSHIFT 6 //Scale up the bias by 26
```

```
#define OUT_RSHIFT 10 //Scale down the output tensor by 1/210
```

```

#define OUT_CH 20
#define OUT_DIM 24
q7_t in_data[3 * IN_DIM * IN_DIM] = {...};
q7_t weight[3 * KER_DIM * KER_DIM * OUT_CH] = {...};
q7_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * 3 * KER_DIM * KER_DIM] = {0};
q7_t out_data[OUT_CH * OUT_DIM * OUT_DIM];

riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias (in_data, IN_DIM,
weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);

```

3.4.3 riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias_fast

Prototype:

```

int32_t riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias_fast (const
q7_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q7_t * bias, const uint16_t bias_lshift, const
uint16_t out_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)

```

Description

This function performs fast signed 8-bit integer convolution for RGB images and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor

- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Example:

//Convolve a 28x28x3 input tensor with a 5x5 kernel and generate a 24x24x20 output

//tensor. Let both dimensions' padding be 0 and their stride be 1.

```
#define IN_DIM 28
```

```
#define KER_DIM 5
```

```
#define PAD 0
```

```
#define STRIDE 1
```

```
#define BIAS_LSHIFT 6 //Scale up the bias by 26
```

```
#define OUT_RSHIFT 10 //Scale down the output tensor by 1/210
```

```
#define OUT_CH 20
```

```
#define OUT_DIM 24
```

```
q7_t in_data[3 * IN_DIM * IN_DIM] = {...};
```

```
q7_t weight[3 * KER_DIM * KER_DIM * OUT_CH] = {...};
```

```
q7_t bias[OUT_CH] = {...};
```

```
q15_t in_tmp_buf[2 * (3 * KER_DIM * KER_DIM + 1)] = {0};
```

```
q15_t wt_tmp_buf[OUT_CH * (3 * KER_DIM * KER_DIM + 1)];
```

```
q7_t out_data[OUT_CH * OUT_DIM * OUT_DIM];
```

```
riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias_fast (in_data,
IN_DIM, weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, wt_tmp_buf);
```

3.4.4 riscv_nn_conv_HWC_s8_s8_s8_sft_bias

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sft_bias (const q7_t
*in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t *ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q7_t *bias, const
uint16_t bias_lshift, const uint16_t out_rshift, q7_t * out_tensor, const
uint16_t out_tensor_dim, q15_t *in_tmp_buf, q7_t *tmp_buf)
```

Description

This function performs signed 8-bit integer convolution and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

0

Example:

```
//Convolve a 28x28x1 input tensor with a 5x5 kernel and generate a
24x24x20
```

```
//output tensor. Let both dimensions' padding be 0 and their stride be
1.
```

```
#define IN_DIM 28
```

```
#define IN_CH 1
```

```
#define KER_DIM 5
```

```
#define PAD 0
```

```
#define STRIDE 1
```

```
#define BIAS_LSHIFT 6 //Scale up the bias by 26
```

```
#define OUT_RSHIFT 10 //Scale down the output tensor by 1/210
```

```

#define OUT_CH 20
#define OUT_DIM 24
q7_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
q7_t weight[IN_CH * KER_DIM * KER_DIM * OUT_CH] = {...};
q7_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * IN_CH * KER_DIM * KER_DIM] = {0};
q7_t out_data[OUT_CH * OUT_DIM * OUT_DIM];

riscv_nn_conv_HWC_s8_s8_s8_sft_bias (in_data, IN_DIM, IN_CH,
weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);

```

3.4.5 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_any

Prototype:

```

void riscv_nn_conv_HWC_s8_s8_s8_sft_bias_any (const q7_t
*in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t *ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q7_t *bias, const uint16_t
bias_lshift, const uint16_t out_rshift, q7_t *out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t *in_tmp_buf,
q7_t *tmp_buf)

```

Description

This function performs signed 8-bit integer convolution in any x and y dimensions and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension

- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value

None

Example:

//Convolve a 160x120x3 input tensor with a 3x5 kernel and generate an 80x59x5

//output tensor. Let both dimensions' padding be 1 and their stride be 2.

```
#define IN_X 160
#define IN_Y 120
#define IN_CH 3
#define OUT_CH 5
#define KER_DIM_X 3
#define KER_DIM_Y 5
#define PAD_X 1
#define PAD_Y 1
#define STRIDE_X 2
#define STRIDE_Y 2
#define BIAS_LSHIFT 6 //Scale up the bias by 26
#define OUT_RSHIFT 9 //Scale down the output tensor by 1/29
#define OUT_X 80
#define OUT_Y 59
q7_t in_data[IN_CH * IN_X * IN_Y] = {...};
```

```

q7_t weight[IN_CH * KER_DIM_X * KER_DIM_Y * OUT_CH] = {...};
q7_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * IN_CH * KER_DIM_X * KER_DIM_Y] = {0};
q7_t out_data[OUT_CH * OUT_X * OUT_Y];

riscv_nn_conv_HWC_s8_s8_s8_sft_bias_any (in_data, IN_X, IN_Y,
IN_CH, weight, OUT_CH, KER_DIM_X, KER_DIM_Y, PAD_X, PAD_Y,
STRIDE_X, STRIDE_Y, bias, BIAS_LSHIFT, OUT_RSHIFT, out_data,
OUT_X, OUT_Y, in_tmp_buf, NULL);

```

3.4.6 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast

Prototype:

```

int32_t riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast (const q7_t
*in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t *ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q7_t *bias, const
uint16_t bias_lshift, const uint16_t out_rshift, q7_t *out_tensor, const
uint16_t out_tensor_dim, q15_t *in_tmp_buf, q7_t *tmp_buf)

```

Description

This function performs fast signed 8-bit integer convolution and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

- [in] *tmp_buf dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2, it returns -1.

Example:

//Convolve a 12x12x20 input tensor with a 5x5 kernel and generate an 8x8x50

//output tensor. Let both dimensions' padding be 0 and their stride be 1.

```
#define IN_DIM 12
```

```
#define IN_CH 20
```

```
#define KER_DIM 5
```

```
#define PAD 0
```

```
#define STRIDE 1
```

```
#define BIAS_LSHIFT 6 //Scale up the bias by 26
```

```
#define OUT_RSHIFT 10 //Scale down the output tensor by 1/210
```

```
#define OUT_CH 50
```

```
#define OUT_DIM 8
```

```
q7_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
```

```
q7_t weight[IN_CH * KER_DIM * KER_DIM * OUT_CH] = {...};
```

```
q7_t bias[OUT_CH] = {...};
```

```
q15_t in_tmp_buf[2 * IN_CH * KER_DIM * KER_DIM] = {0};
```

```
q7_t out_data[OUT_CH * OUT_DIM * OUT_DIM];
```

```
riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast (in_data, IN_DIM,
IN_CH, weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);
```

3.4.7 riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast_any (const q7_t
* in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
```

```
stride_x, const uint16_t stride_y, const q7_t * bias, const uint16_t
bias_lshift, const uint16_t out_rshift, q7_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf,
q7_t * tmp_buf)
```

Description

This function performs signed 8-bit integer convolution in any x and y dimensions and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `**in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2, it returns -1.

Example:

```
//Convolve a 160x120x20 input tensor with a 3x5 kernel and generate
```

an 80x59x8

//output tensor. Let both dimensions' padding be 1 and their stride be 2.

```
#define IN_X 160
#define IN_Y 120
#define IN_CH 20
#define OUT_CH 8
#define KER_DIM_X 3
#define KER_DIM_Y 5
#define PAD_X 1
#define PAD_Y 1
#define STRIDE_X 2
#define STRIDE_Y 2
#define BIAS_LSHIFT 6 //Scale up the bias by 26
#define OUT_RSHIFT 9 //Scale down the output tensor by 1/29
#define OUT_X 80
#define OUT_Y 59
q7_t in_data[IN_CH * IN_X * IN_Y] = {...};
q7_t weight[IN_CH * KER_DIM_X * KER_DIM_Y * OUT_CH] = {...};
q7_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * IN_CH * KER_DIM_X * KER_DIM_Y] = {0};
q7_t out_data[OUT_CH * OUT_Y * OUT_X];

riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast_any (in_data, IN_W,
IN_Y , IN_CH, weight, OUT_CH, KER_DIM_X, KER_DIM_Y, PAD_X,
PAD_Y, STRIDE_X, STRIDE_Y, bias, BIAS_LSHIFT, OUT_RSHIFT,
out_data, OUT_X, OUT_Y, in_tmp_buf, NULL);
```

3.4.8 riscv_nn_conv_HWC_s16_s16_s16_sft_bias

Prototype:

```
int32_t riscv_nn_conv_HWC_s16_s16_s16_sft_bias (const q15_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q15_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q15_t * bias,
const uint16_t bias_lshift, const uint16_t out_rshift, q15_t * out_tensor,
const uint16_t out_tensor_dim, q15_t * in_tmp_buf, q7_t * tmp_buf)
```

Description

This function performs signed 16-bit integer convolution and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

0

Example:

`//Convolve a 28x28x1 input tensor with a 5x5 kernel and generate a 24x24x20`

`//output tensor. Let both dimensions' padding be 0 and their stride be 1.`

`#define IN_DIM 28`

`#define IN_CH 1`

`#define KER_DIM 5`

`#define PAD 0`

`#define STRIDE 1`

`#define BIAS_LSHIFT 6 //Scale up the bias by 26`

`#define OUT_RSHIFT 10 //Scale down the output tensor by 1/210`

```

#define OUT_CH 20
#define OUT_DIM 24
q15_t input_data[IN_CH * IN_DIM * IN_DIM] = {...};
q15_t weight[IN_CH * KER_DIM * KER_DIM * OUT_CH] = {...};
q15_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[IN_CH * KER_DIM * KER_DIM] = {0};
q15_t out_data[OUT_CH * OUT_DIM * OUT_DIM];

riscv_nn_conv_HWC_s16_s16_s16_sft_bias (input_data, IN_DIM,
IN_CH, weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);

```

3.4.9 riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast

Prototype:

```

int32_t riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast (const
q15_t * in_tensor, const uint16_t in_tensor_dim, const uint16_t
in_tensor_ch, const q15_t * ker_weight, const uint16_t out_tensor_ch,
const uint16_t ker_dim, const uint16_t pad, const uint16_t stride, const
q15_t * bias, const uint16_t bias_lshift, const uint16_t out_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf, q7_t *
tmp_buf)

```

Description

This function performs fast signed 16-bit integer convolution and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output

- [out] *out_tensor pointer of the output tensor
- [in] out_tensor_dim dimension of the output tensor
- [in] *in_tmp_buf pointer of temporary buffer for the input tensor
- [in] *tmp_buf dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that both in_tensor_ch and out_tensor_ch should be multiples of 2, it returns -1.

Example:

//Convolve a 28x28x4 input tensor with a 5x5 kernel and generate a 24x24x8 output

//tensor. Let both dimensions' padding be 0 and their stride be 1.

```
#define IN_DIM 28
```

```
#define IN_CH 4
```

```
#define KER_DIM 5
```

```
#define PAD 0
```

```
#define STRIDE 1
```

```
#define BIAS_LSHIFT 6
```

```
#define OUT_RSHIFT 10
```

```
#define OUT_CH 8
```

```
#define OUT_DIM 24
```

```
q15_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
```

```
q15_t weight[IN_CH * KER_DIM * KER_DIM * OUT_CH] = {...};
```

```
q15_t bias[OUT_CH] = {...};
```

```
q15_t in_tmp_buf[IN_CH * KER_DIM * KER_DIM] = {0};
```

```
q15_t out_data[OUT_CH * OUT_DIM * OUT_DIM];
```

```
riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast (in_data, IN_DIM,
IN_CH, weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);
```

3.4.10 riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast_any (const
q15_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
```

```
in_tensor_dim_y, const uint16_t in_tensor_ch, const q15_t * ker_weight,  
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t  
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t  
stride_x, const uint16_t stride_y, const q15_t * bias, const uint16_t  
bias_lshift, const uint16_t out_rshift, q15_t * out_tensor, const uint16_t  
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf,  
q7_t * tmp_buf)
```

Description

This function performs signed 16-bit integer convolution in any x and y dimensions and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that both `in_tensor_ch` and `out_tensor_ch` should be multiples of 2, it returns -1.

Example:

//Convolve a 160x120x20 input tensor with a 3x5 kernel and generate an 80x59x8

//output tensor. Let both dimensions' padding be 1 and their stride be 2.

```
#define IN_X 160
#define IN_Y 120
#define IN_CH 20
#define OUT_CH 8
#define KER_DIM_X 3
#define KER_DIM_Y 5
#define PAD_X 1
#define PAD_Y 1
#define STRIDE_X 2
#define STRIDE_Y 2
#define BIAS_LSHIFT 6 //Scale up the bias by 26
#define OUT_RSHIFT 9 //Scale down the output tensor by 1/29
#define OUT_X 80
#define OUT_Y 59
q15_t in_data[IN_CH * IN_X * IN_Y] = {...};
q15_t weight[IN_CH * KER_DIM_X * KER_DIM_Y * OUT_CH] = {...};
q15_t bias[OUT_CH] = {...};
q15_t in_tmp_buf[2 * IN_CH * KER_DIM_X * KER_DIM_Y] = {0};
q15_t out_data[OUT_CH * OUT_X * OUT_Y];
riscv_nn_conv_HWC_s16_s16_s16_sft_bias_fast_any (in_data,
IN_X, IN_Y, IN_CH, weight, OUT_CH, KER_DIM_X, KER_DIM_Y,
PAD_X, PAD_Y, STRIDE_X, STRIDE_Y, bias, BIAS_LSHIFT,
OUT_RSHIFT, out_data, OUT_X, OUT_Y, in_tmp_buf, NULL);
```

3.4.11 riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias**Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
```

```
ker_dim, const uint16_t pad, const uint16_t stride, const q7_t * bias, const
uint16_t bias_lshift, const uint16_t out_rshift, q7_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q7_t * tmp_buf)
```

Description

This function performs signed 8-bit integer depth-wise convolution and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` equals `out_tensor_ch`, it returns -1.

Example:

```
//Convolve a 11x11x28 input tensor with a 3x3 kernel and generate a
9x9x48 output
//tensor. Let both dimensions' padding be 0 and their stride be 1.
#define IN_DIM 11
#define IN_CH 28
#define OUT_CH 48
```

```

#define KER_DIM 3
#define PAD 0
#define STRIDE 1
#define OUT_RSHIFT 7
#define OUT_DIM 9
q7_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
q7_t weight[IN_CH * KER_DIM * KER_DIM * IN_CH] = {...};
q7_t bias[IN_CH] = {...};
q15_t in_tmp_buf[2 * OUT_CH * KER_DIM * KER_DIM] = {0};
q7_t out_data[OUT_CH * OUT_DIM * OUT_DIM];
riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias (in_data, IN_DIM,
IN_CH, weight, OUT_CH, KER_DIM, PAD, STRIDE, bias, 0,
OUT_RSHIFT, out_data, OUT_DIM, in_tmp_buf, NULL);

```

3.4.12 riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias_any

Prototype:

```

int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias_any (const q7_t
* in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q7_t * bias, const uint16_t
bias_lshift, const uint16_t out_rshift, q7_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf,
q7_t * tmp_buf)

```

Description

This function performs signed 8-bit integer convolution in any x and y dimensions and shift-based quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels

- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Example:

```
//Perform a depth-wise convolution for a 79x59x12 input tensor with a
3x3 kernel

//and generate a 77x57x12 output tensor. Let both dimensions'
padding be 0 and

//their stride be 1.
#define IN_DIM_X 79
#define IN_DIM_Y 59
#define IN_CH 12
#define OUT_CH 12
#define KER_DIM 3
#define PAD 0
#define STRIDE 1
#define BIAS_SHIFT 0
#define OUT_RSHIFT 7
```

```

#define OUT_DIM_X 77
#define OUT_DIM_Y 57
q7_t in_data[IN_CH * IN_DIM_X * IN_DIM_Y] = {...};
q7_t weight[IN_CH * KER_DIM * KER_DIM] = {...};
q7_t bias[IN_CH] = {...};
q15_t in_tmp_buf[2 * OUT_CH * KER_DIM * KER_DIM] = {0};
q7_t out_data[OUT_CH * OUT_DIM_X * OUT_DIM_Y];

riscv_nn_conv_dw_HWC_s8_s8_s8_sft_bias_any (in_data,
IN_DIM_X, IN_DIM_Y, IN_CH, weight, OUT_CH, KER_DIM, KER_DIM,
PAD, PAD, STRIDE, STRIDE, bias, BIAS_SHIFT, OUT_RSHIFT, out_data,
OUT_DIM_X, OUT_DIM_Y, in_tmp_buf, NULL);

```

3.4.13 riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any

Prototype:

```

int32_t riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any
(const q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)

```

Description

This function performs 1x1 kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel

- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. See the note below for detailed information.

Note!

1. The input constraints of this function are:
 - `in_tensor_ch` a multiple of 4
 - `out_tensor_ch` a multiple of 2
 - `ker_dim_x` 1
 - `ker_dim_y` 1
 - `pad_x` 0
 - `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.14 `riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_bias_fast_any`

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_bias_fast_any
(const q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
```

```
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for signed 8-bit integer inputs 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. See the note below for detailed

information.

Note!

1. The input constraints of this function are:
 - `in_tensor_ch` a multiple of 4
 - `out_tensor_ch` a multiple of 2
 - `ker_dim_x` 1
 - `ker_dim_y` 1
 - `pad_x` 0
 - `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.15 riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_bias_fast_any
(const u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights

- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. See the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0
- `pad_y` 0
- `stride_x` 1
- `stride_y` 1

2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.16 riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_bias_fast_any
(const u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor

- `[in] out_tensor_dim_y` y dimension of the output tensor
- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0
- `pad_y` 0
- `stride_x` 1
- `stride_y` 1

2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.17 riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_bias_fast_any**Prototype:**

```
int32_t riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_bias_fast_any
(const u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0

- `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is $out = ((out \gg pre\_rshift) * out\_scale) \gg post\_rshift$.

3.4.18 riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_fast_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const uint16_t pre_rshift, const uint16_t
out_scale, const uint16_t post_rshift, q7_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output

- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:
 - `in_tensor_ch` a multiple of 4
 - `out_tensor_ch` a multiple of 2
 - `ker_dim_x` 1
 - `ker_dim_y` 1
 - `pad_x` 0
 - `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.19 `riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_fast_any`

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_s8_s16_s8_sym_fast_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const uint16_t pre_rshift, const uint16_t
out_scale, const uint16_t post_rshift, q15_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for signed 8-bit integer inputs and 16-bit integer outputs in any x and y dimensions with symmetric

quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0

- `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.20 riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_u8_u8_s8_sym_fast_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const uint16_t pre_rshift, const uint16_t
out_scale, const uint16_t post_rshift, u8_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output

- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. See the note below for detailed information.

Note!

1. The input constraints of this function are:
 - `in_tensor_ch` a multiple of 4
 - `out_tensor_ch` a multiple of 2
 - `ker_dim_x` 1
 - `ker_dim_y` 1
 - `pad_x` 0
 - `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.21 `riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_fast_any`

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_u8_s8_s8_sym_fast_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const uint16_t pre_rshift, const uint16_t
out_scale, const uint16_t post_rshift, q7_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions

with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0

- `pad_y` 0
 - `stride_x` 1
 - `stride_y` 1
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.22 `riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_fast_any`

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_u8_s16_s8_sym_fast_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const uint16_t pre_rshift, const uint16_t
out_scale, const uint16_t post_rshift, q15_t * out_tensor, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs 1x1 kernels convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output

- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. Please see the note below for detailed information.

Note!

1. The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0
- `pad_y` 0
- `stride_x` 1
- `stride_y` 1

2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.23 `riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast`

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast (const
q7_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q31_t * bias, const uint16_t pre_rshift, const
uint16_t out_scale, const uint16_t post_rshift, q7_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for signed 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.24 riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_bias_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_bias_fast (const
q7_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q31_t * bias, const uint16_t pre_rshift, const
uint16_t out_scale, const uint16_t post_rshift, q15_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for signed 8-bit integer inputs 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.25 riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_bias_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_bias_fast (const
u8_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q31_t * bias, const uint16_t pre_rshift, const
uint16_t out_scale, const uint16_t post_rshift, u8_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.26 riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_bias_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_bias_fast (const
u8_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q31_t * bias, const uint16_t pre_rshift, const
uint16_t out_scale, const uint16_t post_rshift, q7_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs and signed 8-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.27 riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_bias_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_bias_fast (const
u8_t * in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad,
const uint16_t stride, const q31_t * bias, const uint16_t pre_rshift, const
uint16_t out_scale, const uint16_t post_rshift, q15_t * out_tensor, const
uint16_t out_tensor_dim, q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs and signed 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.28 riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_fast (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for signed 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.29 riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_RGB_sym_fast (const q7_t
* in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for signed 8-bit integer inputs and 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor

- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.30 riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_RGB_sym_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, u8_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights

- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.31 `riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_fast`

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_RGB_sym_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs and signed 8-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel

- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.32 `riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_fast`

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_RGB_sym_fast (const u8_t
* in_tensor, const uint16_t in_tensor_dim, const q7_t * ker_weight, const
uint16_t out_tensor_ch, const uint16_t ker_dim, const uint16_t pad, const
uint16_t stride, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim,
q15_t * in_tmp_buf, q15_t * wt_tmp_buf)
```

Description

This function performs fast convolution on RGB images for unsigned 8-bit integer inputs and signed 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride

- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [in] `*wt_tmp_buf` pointer of temporary buffer for kernel weights

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.33 riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast (const q7_t
*in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t *ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t *bias, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q7_t *out_tensor, const uint16_t out_tensor_dim, q15_t *in_tmp_buf)
```

Description

This function performs fast convolution on for signed 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector

- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.34 riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast (const q7_t
*in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t *ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t *bias, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q15_t *out_tensor, const uint16_t out_tensor_dim, q15_t *in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs and 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride

- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.35 riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, u8_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size

- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.36 `riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast`

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel

- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.37 riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels

- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.38 riscv_nn_conv_HWC_s8_s8_s8_sym_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sym_fast (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels

- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.39 riscv_nn_conv_HWC_s8_s16_s8_sym_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_sym_fast (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs and 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel

- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.40 riscv_nn_conv_HWC_u8_u8_s8_sym_fast

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_sym_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs/outputs with quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size

- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.41 riscv_nn_conv_HWC_u8_s8_s8_sym_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_sym_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size

- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.42 riscv_nn_conv_HWC_u8_s16_s8_sym_fast**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_sym_fast (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size

- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.43 riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights

- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.44 riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_sym_bias_fast_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs

and 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.45 riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_sym_bias_fast_any (const
```

```
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input

does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.46 riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_sym_bias_fast_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] *bias` pointer of the bias vector
- `[in] pre_rshift` right shift amount for the output before the scaling

- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.47 riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast_any**Prototype:**

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_sym_bias_fast_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels

- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.48 `riscv_nn_conv_HWC_s8_s8_s8_sym_fast_any`

Prototype:

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_sym_fast_any (const q7_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.49 riscv_nn_conv_HWC_s8_s16_s8_sym_fast_any**Prototype:**

```
int32_t riscv_nn_conv_HWC_s8_s16_s8_sym_fast_any (const q7_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
```

```
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs and 16-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.50 riscv_nn_conv_HWC_u8_u8_s8_sym_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_u8_s8_sym_fast_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, u8_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[out] *out_tensor` pointer of the output tensor
- `[in] out_tensor_dim_x` x dimension of the output tensor

- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.51 `riscv_nn_conv_HWC_u8_s8_s8_sym_fast_any`

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s8_s8_sym_fast_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension

- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.52 riscv_nn_conv_HWC_u8_s16_s8_sym_fast_any

Prototype:

```
int32_t riscv_nn_conv_HWC_u8_s16_s8_sym_fast_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs fast convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights

- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 if successful. Otherwise, it returns -1 if the input does not meet the constraints that `in_tensor_ch` must be a multiple of 4 and `out_tensor_ch` must be a multiple of 2.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.53 `riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias`

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.54 riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias**Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer

inputs and 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.55 riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, u8_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.56 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
```

```
post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.57 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
```

```
ker_dim, const uint16_t pad, const uint16_t stride, const q31_t * bias,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim, q15_t *
in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.58 riscv_nn_conv_dw_HWC_s8_s8_s8_sym

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sym (const q7_t *
```

```
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.59 riscv_nn_conv_dw_HWC_s8_s16_s8_sym

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s16_s8_sym (const q7_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
```

```
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs and 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.60 riscv_nn_conv_dw_HWC_u8_u8_s8_sym

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_u8_s8_sym (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
```

```
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.61 riscv_nn_conv_dw_HWC_u8_s8_s8_sym

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s8_s8_sym (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
```

```
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.62 riscv_nn_conv_dw_HWC_u8_s16_s8_sym

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s16_s8_sym (const u8_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const q7_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
```

```
ker_dim, const uint16_t pad, const uint16_t stride, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.63 riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
```

```
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.64 riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias_any**Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_s8_s16_s8_sym_bias_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs and 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling

- [out] *out_tensor pointer of the output tensor
- [in] out_tensor_dim_x x dimension of the output tensor
- [in] out_tensor_dim_y y dimension of the output tensor
- [in] *in_tmp_buf pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that in_tensor_ch is equal to out_tensor_ch, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.65 riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_u8_s8_sym_bias_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] *in_tensor pointer of the input tensor
- [in] in_tensor_dim_x x dimension of the input tensor
- [in] in_tensor_dim_y y dimension of the input tensor
- [in] in_tensor_ch number of input tensor channels
- [in] *ker_weight pointer of kernel weights
- [in] out_tensor_ch number of output tensor channels
- [in] ker_dim_x x dimension of the convolution kernel
- [in] ker_dim_y y dimension of the convolution kernel

- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.66 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s8_s8_sym_bias_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.67 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias_any**Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_u8_s16_s8_sym_bias_any (const
u8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
```

```
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const q31_t * bias, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.68 riscv_nn_conv_dw_HWC_s8_s8_s8_sym_any**Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_sym_any (const q7_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[out] *out_tensor` pointer of the output tensor

- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.69 `riscv_nn_conv_dw_HWC_s8_s16_s8_sym_any`

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s16_s8_sym_any (const q7_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for signed 8-bit integer inputs and 16-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension

- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.70 `riscv_nn_conv_dw_HWC_u8_u8_s8_sym_any`

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_u8_s8_sym_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, u8_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels

- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.71 riscv_nn_conv_dw_HWC_u8_s8_s8_sym_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s8_s8_sym_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q7_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 8-bit integer outputs in any x and y dimensions

with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.72 riscv_nn_conv_dw_HWC_u8_s16_s8_sym_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_u8_s16_s8_sym_any (const u8_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
```

```
const uint16_t in_tensor_ch, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t pre_rshift, const uint16_t out_scale, const
uint16_t post_rshift, q15_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs and signed 16-bit integer outputs in any x and y dimensions with symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.4.73 riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any
(const q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const uint16_t
in_tensor_group, const q7_t * ker_weight, const uint16_t out_tensor_ch,
const uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const int32_t * bias, q7_t * out_tensor, const int32_t *
out_shift, const int32_t * out_scale, const int32_t out_offset, const int32_t
in_offset, const int32_t act_min, const int32_t act_max, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, q15_t * tmp_buf)
```

Description

This function performs fast 1x1 kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `in_tensor_group` number of input tensor groups
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `post_rshift` pointer of the shift vector for the output tensor
- [in] `*out_scale` pointer of the scaling vector for the output tensor
- [in] `out_offset` offset value for the output tensor with the range of -128 to 127

- [\[in\] in_offset](#) offset value for the input tensor with the range of -127 to 128
- [\[in\] act_min](#) minimum value that the output tensor is limited to, and the range is -128 to 127
- [\[in\] act_max](#) maximum value that the output tensor is limited to, and the range is -128 to 127
- [\[in\] out_tensor_dim_x](#) x dimension of the output tensor
- [\[in\] out_tensor_dim_y](#) y dimension of the output tensor
- [\[in\] *tmp_buf](#) dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraints, it returns -1. See the note below for detailed information.

Note!

The input constraints of this function are:

- [in_tensor_ch](#) a multiple of 4
- [pad_x](#) 0
- [pad_y](#) 0
- [stride_x](#) 1
- [stride_y](#) 1

3.4.74**riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size****Prototype:**

```
int32_t
riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size
(const uint16_t in_tensor_ch)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of [riscv_nn_conv_1x1_HWC_s8_s8_s8_asym_bias_fast_any](#).

Parameter:

- [\[in\] in_tensor_ch](#) number of input tensor channels

Return Value:

This function returns the size required by the temporary buffer.

3.4.75 riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any**Prototype:**

```
int riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any (const q7_t
* in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_ch,
const uint16_t in_tensor_group, const q7_t * ker_weight, const uint16_t
out_tensor_ch, const uint16_t ker_dim_x, const uint16_t pad_x, const
uint16_t stride_x, const int32_t * bias, q7_t * out_tensor, const int32_t *
out_shift, const int32_t * out_scale, const int32_t out_offset, const int32_t
in_offset, const int32_t act_min, const int32_t act_max, const uint16_t
out_tensor_dim_x, q15_t * in_tmp_buf)
```

Description

This function performs 1xn kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `in_tensor_group` dummy
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `post_rshift` pointer of the shift vector for the output tensor
- [in] `*out_scale` pointer of the scaling vector for the output tensor
- [in] `out_offset` offset value for the output tensor with the range of -128 to 127
- [in] `in_offset` offset value for the input tensor with the range of -127 to 128

- [\[in\] act_min](#) minimum value that the output tensor is limited to, and the range is -128 to 127
- [\[in\] act_max](#) maximum value that the output tensor is limited to, and the range is -128 to 127
- [\[in\] out_tensor_dim_x](#) x dimension of the output tensor
- [\[in\] *tmp_buf](#) point of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that [in_tensor_dim_x](#) should be multiples of 4, it returns -1.

3.4.76**riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any_get_buffer_size****Prototype:**

```
int32_t
riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any_get_buffer_size
(const uint16_t in_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of

```
riscv_nn_conv_1xn_HWC_s8_s8_s8_asym_bias_any.
```

Parameter:

- [\[in\] in_tensor_ch](#) number of input tensor channels
- [\[in\] ker_dim_x](#) x dimension of the convolution kernel
- [\[in\] ker_dim_y](#) y dimension of the convolution kernel, and it is always 1

Return Value:

This function returns the size required by the temporary buffer.

3.4.77 riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any**Prototype:**

```
int32_t riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any (const q7_t *
in_tensor, const uint16_t in_tensor_dim_x, const uint16_t in_tensor_dim_y,
const uint16_t in_tensor_ch, const uint16_t in_tensor_group, const q7_t *
ker_weight, const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const
```

```
uint16_t ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const
uint16_t stride_x, const uint16_t stride_y, const int32_t * bias, q7_t *
out_tensor, const int32_t * out_shift, const int32_t * out_scale, const
int32_t out_offset, const int32_t in_offset, const int32_t act_min, const
int32_t act_max, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q15_t * in_tmp_buf)
```

Description

This function performs convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `in_tensor_group` number of input tensor groups
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_rshift` pointer of the shift vector for the output tensor
- [in] `*out_scale` pointer of the scaling vector for the output tensor
- [in] `out_offset` offset value for the output tensor with the range of -128 to 127
- [in] `in_offset` offset value for the input tensor with the range of -127 to 128
- [in] `act_min` minimum value that the output tensor is limited to, and the range is -128 to 127
- [in] `act_max` maximum value that the output tensor is limited to, and

the range is -128 to 127

- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

3.4.78 `riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any_get_buffer_size`

Prototype:

```
int32_t
riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any_get_buffer_size (const
uint16_t in_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of `riscv_nn_conv_HWC_s8_s8_s8_asym_bias_any`.

Parameter:

- [in] `in_tensor_ch` number of input tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel

Return Value:

This function returns the size required by the temporary buffer.

3.4.79 `riscv_nn_conv_dw_HWC_3x3_s8_s8_s8_asym_bias_any`

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_3x3_s8_s8_s8_asym_bias_any
(const int8_t * in_tensor, const int32_t in_tensor_dim_x, const int32_t
in_tensor_dim_y, const int32_t in_tensor_ch, const int8_t * ker_weight,
const int32_t out_tensor_ch, const int32_t pad_x, const int32_t pad_y,
const int32_t stride_x, const int32_t stride_y, const int32_t * bias, int8_t *
out_tensor, const int32_t * out_shift, const int32_t * out_scale, const
int32_t out_tensor_dim_x, const int32_t out_tensor_dim_y, const int32_t
out_offset, const int32_t in_offset, const int32_t act_min, const int32_t
act_max, const int32_t dilation_x, const int32_t dilation_y, int16_t *
tmp_buf)
```

Description

This function performs depth-wise 3x3 kernels convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_rshift` pointer of the shift vector for the output tensor
- [in] `*out_scale` pointer of the scaling vector for the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `out_offset` offset value for the output tensor with the range of -128 to 127
- [in] `in_offset` offset value for the input tensor with the range of -127 to 128
- [in] `act_min` minimum value that the output tensor is limited to, and the range is -128 to 127
- [in] `act_max` maximum value that the output tensor is limited to, and the range is -128 to 127
- [in] `dilation_x` dummy
- [in] `dilation_y` dummy
- [in] `*tmp_buf` dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not

satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch` and `pad_x` is less than 1, it returns -1.

3.4.80 riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_any (const
q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ch_mult, const uint16_t
ker_dim_x, const uint16_t ker_dim_y, const uint16_t pad_x, const uint16_t
pad_y, const uint16_t stride_x, const uint16_t stride_y, const int32_t * bias,
q7_t * out_tensor, const int32_t * out_shift, const int32_t * out_scale, const
uint16_t out_tensor_dim_x, const uint16_t out_tensor_dim_y, const int32_t
out_offset, const int32_t in_offset, const int32_t act_min, const int32_t
act_max, const uint16_t dilation_x, const uint16_t dilation_y, q15_t *
tmp_buf)
```

Description

This function performs convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels, equal to "ch_mult * in_tensor_ch"
- `[in] ch_mult` multiplier of input tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] pad_x` padding size in the x dimension
- `[in] pad_y` padding size in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] stride_y` convolution stride in the y dimension
- `[in] *bias` pointer of the bias vector

- [out] *out_tensor pointer of the output tensor
- [in] out_rshift pointer of the shift vector for the output tensor
- [in] * out_scale pointer of the scaling vector for the output tensor
- [in] out_tensor_dim_x x dimension of the output tensor
- [in] out_tensor_dim_y y dimension of the output tensor
- [in] out_offset offset value for the output tensor with the range of -128 to 127
- [in] in_offset offset value for the input tensor with the range of -127 to 128
- [in] act_min minimum value that the output tensor is limited to, and the range is -128 to 127
- [in] act_max maximum value that the output tensor is limited to, and the range is -128 to 127
- [in] dilation_x dummy
- [in] dilation_y dummy
- [in] *tmp_buf dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that in_tensor_ch equals out_tensor_ch, it returns -1.

3.4.81 riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any
(const q7_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const q7_t * ker_weight,
const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const uint16_t
stride_x, const uint16_t stride_y, const int32_t * bias, q7_t * out_tensor,
const int32_t * out_shift, const int32_t * out_scale, const uint16_t
out_tensor_dim_x, const uint16_t out_tensor_dim_y, const int32_t
out_offset, const int32_t in_offset, const int32_t act_min, const int32_t
act_max, const uint16_t dilation_x, const uint16_t dilation_y, q15_t *
in_tmp_buf)
```

Description

This function performs fast convolution for signed 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_rshift` pointer of the shift vector for the output tensor
- [in] `*out_scale` pointer of the scaling vector for the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `out_offset` offset value for the output tensor with the range of -128 to 127
- [in] `in_offset` offset value for the input tensor with the range of -127 to 128
- [in] `act_min` minimum value that the output tensor is limited to, and the range is -128 to 127
- [in] `act_max` maximum value that the output tensor is limited to, and the range is -128 to 127
- [in] `dilation_x` dummy
- [in] `dilation_y` dummy
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

3.4.82

riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size**Prototype:**

```
int32_t
riscv_nn_conv_dw_HWC_s8_s8_s8_asym_bias_fast_any_get_buffer_size
(const uint16_t in_tensor_ch, const uint16_t ker_dim_x, const uint16_t
ker_dim_y)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of `riscv_nn_conv_dw_HWC_s8_fast_any`.

Parameter:

- `[in] in_tensor_ch` number of input tensor channels
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] ker_dim_y` y dimension of the convolution kernel

Return Value:

This function returns the size required by the temporary buffer.

3.4.83 **riscv_nn_conv_dw_HWC_u8_u8_u8_asym_bias_any****Prototype:**

```
int32_t riscv_nn_conv_dw_HWC_u8_u8_u8_asym_bias_any (const
uint8_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const uint8_t * ker_weight,
const uint16_t ker_dim_x, const uint16_t ker_dim_y, const int16_t ch_mult,
const int16_t pad_x, const int16_t pad_y, const int16_t stride_x, const
int16_t stride_y, const int16_t dilation_x, const int16_t dilation_y, const
int32_t * bias, const int32_t in_offset, const int32_t ker_offset, const
int32_t out_offset, uint8_t * out_tensor, const uint16_t out_tensor_dim_x,
const uint16_t out_tensor_dim_y, const int32_t act_min, const int32_t
act_max, const int32_t out_shift, const int32_t out_scale)
```

Description

This function performs depth-wise convolution for unsigned 8-bit integer inputs/outputs in any x and y dimensions with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `ch_mult` multiplier of input tensor channels
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `dilation_x` dummy
- [in] `dilation_y` dummy
- [in] `*bias` pointer of the bias vector
- [in] `in_offset` offset value for the input tensor with the range of -255 to 0
- [in] `ker_offset` offset value for the convolution kernel with the range of -255 to 0
- [in] `out_offset` offset value for the output tensor with the range of 0 to -255
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `act_min` minimum value that the output tensor is limited to, and the range is 0 to 255
- [in] `act_max` maximum value that the output tensor is limited to, and the range is 0 to 255
- [in] `out_rshift` shift amount for the output tensor
- [in] `out_scale` scaling value for output tensor

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that both `ch_mult` and `ker_dim_x` are multiple of

2, it returns -1.

3.4.84 riscv_nn_conv_1x1_HWC_f16_f16_f16_bias_any

Prototype:

```
int32_t riscv_nn_conv_1x1_HWC_f16_f16_f16_bias_any (const
float16_t * in_tensor, const uint16_t in_tensor_dim_x, const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_ch, const float16_t *
ker_weight, const uint16_t out_tensor_ch, const uint16_t ker_dim_x, const
uint16_t ker_dim_y, const uint16_t pad_x, const uint16_t pad_y, const
uint16_t stride_x, const uint16_t stride_y, const float16_t * bias, float16_t *
out_tensor, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, float16_t * in_tmp_buf, float16_t * tmp_buf)
```

Description

This function performs 1x1 kernels convolution for half-precision floating-point inputs/outputs in any x and y dimensions.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*ker_weight` pointer of kernel weights
- [in] `out_tensor_ch` number of output tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `*bias` pointer of the bias vector
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` dummy pointer
- [in] `*tmp_buf` dummy pointer

Return Value:

0

Note!

The input constraints of this function are:

- `in_tensor_ch` a multiple of 4
- `out_tensor_ch` a multiple of 2
- `ker_dim_x` 1
- `ker_dim_y` 1
- `pad_x` 0
- `pad_y` 0
- `stride_x` 1
- `stride_y` 1

3.4.85 riscv_nn_conv_HWC_f16_f16_f16_bias**Prototype:**

```
int32_t riscv_nn_conv_HWC_f16_f16_f16_bias (const float16_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const float16_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const float16_t * bias,
float16_t * out_tensor, const uint16_t out_tensor_dim, float16_t *
in_tmp_buf, float16_t * tmp_buf)
```

Description

This function performs convolution for half-precision floating-point inputs/outputs.

Parameter:

- `[in] *in_tensor` pointer of the input tensor
- `[in] in_tensor_dim` dimension of the input tensor
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *ker_weight` pointer of kernel weights
- `[in] out_tensor_ch` number of output tensor channels
- `[in] ker_dim` dimension of the convolution kernel
- `[in] pad` padding size
- `[in] stride` convolution stride
- `[in] *bias` pointer of the bias vector

- [out] *out_tensor pointer of the output tensor
- [in] out_tensor_dim dimension of the output tensor
- [in] *in_tmp_buf pointer of temporary buffer for the input tensor
- [in] *tmp_buf dummy pointer

Return Value:

0

3.4.86 riscv_nn_conv_dw_HWC_f16_f16_f16_bias

Prototype:

```
int32_t riscv_nn_conv_dw_HWC_f16_f16_f16_bias (const float16_t *
in_tensor, const uint16_t in_tensor_dim, const uint16_t in_tensor_ch,
const float16_t * ker_weight, const uint16_t out_tensor_ch, const uint16_t
ker_dim, const uint16_t pad, const uint16_t stride, const float16_t * bias,
float16_t * out_tensor, const uint16_t out_tensor_dim, float16_t *
in_tmp_buf, float16_t * tmp_buf)
```

Description

This function performs depth-wise convolution for half-precision floating-point inputs/outputs.

Parameter:

- [in] *in_tensor pointer of the input tensor
- [in] in_tensor_dim dimension of the input tensor
- [in] in_tensor_ch number of input tensor channels
- [in] *ker_weight pointer of kernel weights
- [in] out_tensor_ch number of output tensor channels
- [in] ker_dim dimension of the convolution kernel
- [in] pad padding size
- [in] stride convolution stride
- [in] *bias pointer of the bias vector
- [out] *out_tensor pointer of the output tensor
- [in] out_tensor_dim dimension of the output tensor
- [in] *in_tensor pointer of temporary buffer for the input tensor
- [in] *tmp_buf dummy pointer

Return Value:

The function returns 0 on success. Otherwise, if the input does not satisfy the constraint that `in_tensor_ch` is equal to `out_tensor_ch`, it returns -1.

3.5 Fully Connected Functions

The fully connected functions multiply an input vector by a weight matrix and adds a bias (if any) to the result. Supported combinations for the input vector and weight matrix include (signed 8-bit integer, signed 8-bit integer), (unsigned 8-bit integer, signed 8-bit integer), (signed 16-bit integer, signed 8-bit integer), (signed 16-bit integer, signed 16-bit integer), and (half-precision floating point, half-precision floating point).

3.5.1 riscv_nn_fc_s8_s8_s8_sft_bias

Prototype:

```
int32_t riscv_nn_fc_s8_s8_s8_sft_bias (const q7_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t bias_lshift, const uint16_t out_rshift, const q7_t * bias, q7_t *
out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs with shift-based quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] bias_lshift` left shift amount for the bias
- `[in] out_rshift` right shift amount for the output
- `[in] *bias` pointer of the bias vector
- `[out] *out_vec` pointer of the output vector
- `[in] *in_tmp_buf` dummy pointer

Return Value:

0

Example:

```
#define IN_SIZE 2048
#define OUT_SIZE 256
```

```

#define BIAS_LSHIFT 9 //Scale up the bias by 29
#define OUT_RSHIFT 9 //Scale down the outputs by 1/29
q7_t in_vec[IN_SIZE] = {...};
q7_t wt_mat[IN_SIZE * OUT_SIZE] {...};
q7_t bias[OUT_SIZE] = {...};
q7_t out_vec[OUT_SIZE];

riscv_nn_fc_s8_s8_s8_sft_bias (in_vec, wt_mat, IN_SIZE,
OUT_SIZE, BIAS_LSHIFT, OUT_RSHIFT, bias, out_vec, NULL);

```

3.5.2 riscv_nn_fc_s8_s8_s8_sft_bias_fast

Prototype:

```

int32_t riscv_nn_fc_s8_s8_s8_sft_bias_fast (const q7_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t bias_lshift, const uint16_t out_rshift, const q7_t * bias, q7_t
* out_vec, q15_t * in_tmp_buf)

```

Description

This is a fully connected layer function for signed 8-bit integer inputs with interleaved multiplication and shift-based quantization on the outputs.

Parameter:

- [in] *in_vec* pointer of the input vector
- [in] *wt_mat* pointer of the weight matrix
- [in] *size* number of elements in the input vector
- [in] *wt_row_num* number of rows in the weight matrix
- [in] *bias_lshift* left shift amount for the bias
- [in] *out_rshift* right shift amount for the output
- [in] **bias* pointer of the bias vector
- [out] **out_vec* pointer of the output vector
- [in] **in_tensor* pointer of temporary buffer for the input tensor

Return Value:

0

Note!

In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s8_wt_converter](#).

3.5.3 riscv_nn_fc_s16_s16_s16_sft_bias

Prototype:

```
int32_t riscv_nn_fc_s16_s16_s16_sft_bias (const q15_t * in_vec,  
const q15_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,  
const uint16_t bias_lshift, const uint16_t out_rshift, const q15_t * bias,  
q15_t * out_vec, q15_t * tmp_buf)
```

Description

This is a fully connected layer function for signed 16-bit integer inputs with shift-based quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*tmp_buf` dummy pointer

Return Value:

0

3.5.4 riscv_nn_fc_s16_s16_s16_sft_bias_fast

Prototype:

```
int32_t riscv_nn_fc_s16_s16_s16_sft_bias_fast (const q15_t * in_vec,  
const q15_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,  
const uint16_t bias_lshift, const uint16_t out_rshift, const q15_t * bias,  
q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 16-bit integer inputs with interleaved multiplication and shift-based quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix

- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input vector

Return Value:

0

Note!

In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s16_wt_converter](#).

3.5.5 riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias

Prototype:

```
int32_t riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias (const q15_t *
in_vec, const q7_t * wt_mat, const uint16_t size, const uint16_t
wt_row_num, const uint16_t bias_lshift, const uint16_t out_rshift, const
q7_t * bias, q15_t * out_vec, q15_t * tmp_buf)
```

Description

This function multiplies a signed 16-bit integer input vector by a signed 8-bit integer weight matrix and performs shift-based quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `bias_lshift` left shift amount for the bias
- [in] `out_rshift` right shift amount for the output
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*tmp_buf` dummy pointer

Return Value:

0

3.5.6 riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias_fast**Prototype:**

```
int32_t riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias_fast (const q15_t
* in_vec, const q7_t * wt_mat, const uint16_t size, const uint16_t
wt_row_num, const uint16_t bias_lshift, const uint16_t out_rshift, const
q7_t * bias, q15_t * out_vec, q15_t * tmp_buf)
```

Description

This function multiplies a signed 16-bit integer input vector by a signed 8-bit integer weight matrix in interleaved format and performs shift-based quantization on the outputs.

Parameter:

- [in] *in_vec* pointer of the input vector
- [in] *wt_mat* pointer of the weight matrix
- [in] *size* number of elements in the input vector
- [in] *wt_row_num* number of rows in the weight matrix
- [in] *bias_lshift* left shift amount for the bias
- [in] *out_rshift* right shift amount for the output
- [in] **bias* pointer of the bias vector
- [out] **out_vec* pointer of the output vector
- [in] **tmp_buf* dummy pointer

Return Value:

0

Note!

In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_mat_vec_s8_wt_converter](#).

3.5.7 riscv_nn_fc_s8_s8_s8_sym_bias**Prototype:**

```
int32_t riscv_nn_fc_s8_s8_s8_sym_bias (const q7_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
const q31_t * bias, q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.8 riscv_nn_fc_s8_s16_s8_sym_bias**Prototype:**

```
int32_t riscv_nn_fc_s8_s16_s8_sym_bias (const q7_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
const q31_t * bias, q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs and 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector

- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.9 riscv_nn_fc_u8_u8_s8_sym_bias

Prototype:

```
int32_t riscv_nn_fc_u8_u8_s8_sym_bias (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
const q31_t * bias, u8_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.10 riscv_nn_fc_u8_s8_s8_sym_bias**Prototype:**

```
int32_t riscv_nn_fc_u8_s8_s8_sym_bias (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
const q31_t * bias, q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 8-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[in] *bias` pointer of the bias vector
- `[out] *out_vec` pointer of the output vector
- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.11 riscv_nn_fc_u8_s16_s8_sym_bias

Prototype:

```
int32_t riscv_nn_fc_u8_s16_s8_sym_bias (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
const q31_t * bias, q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 16-bit integer outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] *in_vec* pointer of the input vector
- [in] *wt_mat* pointer of the weight matrix
- [in] *size* number of elements in the input vector
- [in] *wt_row_num* number of rows in the weight matrix
- [in] *pre_rshift* right shift amount for the output before the scaling
- [in] *out_scale* scaling value for the output
- [in] *post_rshift* right shift amount for the output after the scaling
- [in] **bias* pointer of the bias vector
- [out] **out_vec* pointer of the output vector
- [in] **in_tmp_buf* pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.12 riscv_nn_fc_s8_s8_s8_sym

Prototype:

```
int32_t riscv_nn_fc_s8_s8_s8_sym (const q7_t * in_vec, const q7_t *
wt_mat, const uint16_t size, const uint16_t wt_row_num, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[out] *out_vec` pointer of the output vector
- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.13 riscv_nn_fc_s8_s16_s8_sym

Prototype:

```
int32_t riscv_nn_fc_s8_s16_s8_sym (const q7_t * in_vec, const q7_t *
wt_mat, const uint16_t size, const uint16_t wt_row_num, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs and 16-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] pre_rshift` right shift amount for the output before the scaling

- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.14 `riscv_nn_fc_u8_u8_s8_sym`

Prototype:

```
int32_t riscv_nn_fc_u8_u8_s8_sym (const u8_t * in_vec, const q7_t *
wt_mat, const uint16_t size, const uint16_t wt_row_num, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, u8_t *
out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs/outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.15 riscv_nn_fc_u8_s8_s8_sym

Prototype:

```
int32_t riscv_nn_fc_u8_s8_s8_sym (const u8_t * in_vec, const q7_t *
wt_mat, const uint16_t size, const uint16_t wt_row_num, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q7_t *
out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 8-bit integer outputs with symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.16 riscv_nn_fc_u8_s16_s8_sym

Prototype:

```
int32_t riscv_nn_fc_u8_s16_s8_sym (const u8_t * in_vec, const q7_t
* wt_mat, const uint16_t size, const uint16_t wt_row_num, const uint16_t
pre_rshift, const uint16_t out_scale, const uint16_t post_rshift, q15_t *
out_vec, q15_t * in_tmp_buf);
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 16-bit integer outputs with symmetric quantization on

the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.17 riscv_nn_fc_s8_s8_s8_sym_bias_fast

Prototype:

```
int32_t riscv_nn_fc_s8_s8_s8_sym_bias_fast (const q7_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, const q31_t * bias, q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs/outputs with bias inputs, interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output

- [in] `post_rshift` right shift amount for the output after the scaling
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.18 `riscv_nn_fc_s8_s16_s8_sym_bias_fast`

Prototype:

```
int32_t riscv_nn_fc_s8_s16_s8_sym_bias_fast (const q7_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, const q31_t * bias, q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs and 16-bit integer outputs with bias inputs, interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [in] `*bias` pointer of the bias vector
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s8_wt_converter](#).
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.19 riscv_nn_fc_u8_u8_s8_sym_bias_fast**Prototype:**

```
int32_t riscv_nn_fc_u8_u8_s8_sym_bias_fast (const u8_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, const q31_t * bias, u8_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs/outputs with bias inputs, interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[in] *bias` pointer of the bias vector
- `[out] *out_vec` pointer of the output vector
- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s8_wt_converter](#).
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.20 riscv_nn_fc_u8_s8_s8_sym_bias_fast

Prototype:

```
int32_t riscv_nn_fc_u8_s8_s8_sym_bias_fast (const u8_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, const q31_t * bias, q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 8-bit integer outputs with bias inputs, interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [in] *in_vec* pointer of the input vector
- [in] *wt_mat* pointer of the weight matrix
- [in] *size* number of elements in the input vector
- [in] *wt_row_num* number of rows in the weight matrix
- [in] *pre_rshift* right shift amount for the output before the scaling
- [in] *out_scale* scaling value for the output
- [in] *post_rshift* right shift amount for the output after the scaling
- [in] **bias* pointer of the bias vector
- [out] **out_vec* pointer of the output vector
- [in] **in_tmp_buf* pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s8_wt_converter](#).
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.21 riscv_nn_fc_u8_s16_s8_sym_bias_fast

Prototype:

```
int32_t riscv_nn_fc_u8_s16_s8_sym_bias_fast (const u8_t * in_vec,
const q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num,
const uint16_t pre_rshift, const uint16_t out_scale, const uint16_t
post_rshift, const q31_t * bias, q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 16-bit integer outputs with bias inputs, interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [\[in\] *in_vec](#) pointer of the input vector
- [\[in\] *wt_mat](#) pointer of the weight matrix
- [\[in\] size](#) number of elements in the input vector
- [\[in\] wt_row_num](#) number of rows in the weight matrix
- [\[in\] pre_rshift](#) right shift amount for the output before the scaling
- [\[in\] out_scale](#) scaling value for the output
- [\[in\] post_rshift](#) right shift amount for the output after the scaling
- [\[in\] *bias](#) pointer of the bias vector
- [\[out\] *out_vec](#) pointer of the output vector
- [\[in\] *in_tmp_buf](#) pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from [riscv_nn_fc_s8_wt_converter](#).
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.22 riscv_nn_fc_s8_s8_s8_sym_fast**Prototype:**

```
int32_t riscv_nn_fc_s8_s8_s8_sym_fast (const q7_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs/outputs with interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [\[in\] *in_vec](#) pointer of the input vector

- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.23 `riscv_nn_fc_s8_s16_s8_sym_fast`

Prototype:

```
int32_t riscv_nn_fc_s8_s16_s8_sym_fast (const q7_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs and 16-bit integer outputs with interleaved multiplication, and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector

- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.24 riscv_nn_fc_u8_u8_s8_sym_fast

Prototype:

```
int32_t riscv_nn_fc_u8_u8_s8_sym_fast (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
u8_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs/outputs with interleaved multiplication and symmetric quantization on the outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] pre_rshift` right shift amount for the output before the scaling
- `[in] out_scale` scaling value for the output
- `[in] post_rshift` right shift amount for the output after the scaling
- `[out] *out_vec` pointer of the output vector
- `[in] *in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.25 riscv_nn_fc_u8_s8_s8_sym_fast

Prototype:

```
int32_t riscv_nn_fc_u8_s8_s8_sym_fast (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q7_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 8-bit integer outputs with interleaved multiplication and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.26 riscv_nn_fc_u8_s16_s8_sym_fast

Prototype:

```
int32_t riscv_nn_fc_u8_s16_s8_sym_fast (const u8_t * in_vec, const
q7_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
uint16_t pre_rshift, const uint16_t out_scale, const uint16_t post_rshift,
q15_t * out_vec, q15_t * in_tmp_buf)
```

Description

This is a fully connected layer function for unsigned 8-bit integer inputs and signed 16-bit integer outputs with interleaved multiplication and symmetric quantization on the outputs.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [in] `pre_rshift` right shift amount for the output before the scaling
- [in] `out_scale` scaling value for the output
- [in] `post_rshift` right shift amount for the output after the scaling
- [out] `*out_vec` pointer of the output vector
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor

Return Value:

0

Note!

1. In this function, the input vector is multiplied by a weight matrix in interleaved format obtained from `riscv_nn_fc_s8_wt_converter`.
2. The outputs will be two-stage shifted before being stored, that is `out = ((out >> pre_rshift) * out_scale) >> post_rshift`.

3.5.27 riscv_nn_fc_s8_wt_converter

Prototype:

```
void riscv_nn_fc_s8_wt_converter (const q7_t * wt_mat, const
uint32_t size, const uint32_t wt_row_num, q7_t * wt_mat_out)
```

Description

This is a weight converter for fully-connected functions with signed 8-bit weight data and named with "fast".

Parameter:

- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [out] `*wt_mat_out` pointer of the weight matrix stored in a specific order

Return Value:

None

3.5.28 riscv_nn_fc_s16_wt_converter

Prototype:

```
void riscv_nn_fc_s16_wt_converter (const q15_t * wt_mat, const
uint32_t size, const uint32_t wt_row_num, q15_t * wt_mat_out)
```

Description

This is a weight converter for fully-connected functions with signed 16-bit weight data and named with "fast".

Parameter:

- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [out] `*wt_mat_out` pointer of the weight matrix stored in a specific order

Return Value:

None

3.5.29 riscv_nn_fc_mat_vec_s8_wt_converter

Prototype:

```
void riscv_nn_fc_mat_vec_s8_wt_converter (const q7_t * wt_mat,
const uint32_t size, const uint32_t wt_row_num, q7_t * wt_mat_out)
```

Description

This is a weight converter for `riscv_nn_fc_mat_vec_s16_s16_s8_sft_bias_fast`.

Parameter:

- [in] `*wt_mat` pointer of the weight matrix
- [in] `size` number of elements in the input vector
- [in] `wt_row_num` number of rows in the weight matrix
- [out] `*wt_mat_out` pointer of the weight matrix stored in a specific order

Return Value:

None

3.5.30 riscv_nn_fc_s8_s8_s8_asym_bias

Prototype:

```
int32_t riscv_nn_fc_s8_s8_s8_asym_bias (const int8_t * in_vec, const
int8_t * wt_mat, const uint16_t in_vec_col, const uint16_t wt_mat_row,
const uint16_t in_vec_group, const int32_t in_offset, const int32_t
wt_offset, const int32_t out_scale, const int32_t out_shift, const int32_t
out_offset, const int32_t * bias, int8_t * out_vec, const int32_t act_min,
const int32_t act_max, q15_t * tmp_buf)
```

Description

This is a fully connected layer function for signed 8-bit integer inputs/outputs with bias inputs and symmetric quantization on the outputs.

Parameter:

- [in] **in_vec* pointer of the input vector
- [in] **wt_mat* pointer of the weight matrix
- [in] *in_vec_col* number of columns in the input vector (or transposed weight matrix)
- [in] *wt_mat_row* number of rows in the transposed weight matrix
- [in] *in_vec_group* number of input vector groups
- [in] *in_offset* offset value to be added to the input vector with the range of -127 to 128
- [in] *out_offset* offset value to be added to the weight with the range of -127 to 128
- [in] *out_scale* scaling value for the output vector
- [in] *out_rshift* shift amount for the output vector
- [in] *out_offset* offset value to be added to the output vector with the range of -128 to 127
- [in] **bias* pointer of the bias vector
- [out] **out_vec* pointer of the output vector
- [in] *act_min* minimum value that the output vector is limited to, and the range is -128 to 127
- [in] *act_max* maximum value that the output vector is limited to, and the range is -128 to 127
- [in] **tmp_buf* dummy pointer

Return Value:

0

3.5.31 riscv_nn_fc_s8_s8_s8_asym_bias_get_buffer_size

Prototype:

```
int32_t riscv_nn_fc_s8_s8_s8_asym_bias_get_buffer_size (const
uint16_t in_vec_col)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of `riscv_nn_fc_s8_s8_s8_asym_bias`.

Parameter:

- `[in] in_vec_col` number of columns in the input vector (or transposed weight matrix)

Return Value:

This function returns the size required by the temporary buffer.

3.5.32 riscv_nn_fc_f16_f16_f16_bias

Prototype:

```
int32_t riscv_nn_fc_f16_f16_f16_bias (const float16_t * in_vec, const
float16_t * wt_mat, const uint16_t size, const uint16_t wt_row_num, const
float16_t * bias, float16_t * out_vec, float16_t * tmp_buf)
```

Description

This is a fully connected layer function for half-precision floating-point inputs/outputs.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] *wt_mat` pointer of the weight matrix
- `[in] size` number of elements in the input vector
- `[in] wt_row_num` number of rows in the weight matrix
- `[in] *bias` pointer of the bias vector
- `[out] *out_vec` pointer of the output vector
- `[in] *tmp_buf` dummy pointer

Return Value:

0

3.6 Pooling Functions

The pooling functions are used for downsampling input data and

include max pooling and average pooling functions.

3.6.1 riscv_nn_avepool_HWC_s8

Prototype:

```
void riscv_nn_avepool_HWC_s8 (q7_t * in_tensor, const uint16_t
in_tensor_dim, const uint16_t in_tensor_ch, const uint16_t ker_dim, const
uint16_t pad, const uint16_t stride, const uint16_t out_tensor_dim, q7_t *
in_tmp_buf, q7_t * out_tensor)
```

Description

This is an average pooling function for signed 8-bit integer inputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [out] `*out_tensor` pointer of the output tensor

Return Value:

None

Example:

```
#define IN_DIM 32
#define IN_CH 32
#define KER_DIM 3
#define PAD 0
#define STRIDE 2
#define OUT_DIM 15
q7_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
q7_t out_data[IN_CH * OUT_DIM * OUT_DIM] = {...};
q7_t in_tmp_buf[2 * OUT_DIM * IN_CH];
riscv_nn_avepool_HWC_s8 (in_data, IN_DIM, IN_CH, KER_DIM,
PAD, STRIDE, OUT_DIM, in_tmp_buf, out_data);
```

3.6.2 riscv_nn_avepool_HWC_s8_any

Prototype:

```
void riscv_nn_avepool_HWC_s8_any (q7_t * in_tensor, const uint16_t
in_tensor_dim_x, const uint16_t in_tensor_dim_y, const uint16_t
in_tensor_ch, const uint16_t ker_dim_x, const uint16_t ker_dim_y, const
uint16_t pad_x, const uint16_t pad_y, const uint16_t stride_x, const
uint16_t stride_y, const uint16_t out_tensor_dim_x, const uint16_t
out_tensor_dim_y, q7_t * in_tmp_buf, q7_t * out_tensor, const uint16_t
out_lshift)
```

Description

This is an average pooling function for signed 8-bit integer inputs in any x and y dimensions.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_ch` The number of input tensor channels
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `pad_x` padding size in the x dimension
- [in] `pad_y` padding size in the y dimension
- [in] `stride_x` convolution stride in the x dimension
- [in] `stride_y` convolution stride in the y dimension
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [out] `*out_tensor` pointer of the output tensor
- [in] `out_rshift` left shift amount for the output

Return Value:

None

Example:

```
#define IN_X 160
#define IN_Y 120
```

```

#define IN_CH 3
#define KER_DIM_X 3
#define KER_DIM_Y 5
#define PAD_X 1
#define PAD_Y 1
#define STRIDE_X 2
#define STRIDE_Y 2
#define OUT_LSHIFT 3
#define OUT_X 80
#define OUT_Y 59
q7_t in_data[IN_CH * IN_X * IN_Y] = {...};
q7_t out_data[IN_CH * OUT_X * OUT_Y] = {...};
q7_t in_tmp_buf[2 * IN_CH * OUT_X * OUT_Y];
riscv_nn_avepool_HWC_s8_any (in_data, IN_X, IN_Y, IN_CH,
KER_DIM_X, KER_DIM_Y, PAD_X, PAD_Y, STRIDE_X, STRIDE_Y,
OUT_X, OUT_Y, in_tmp_buf, out_data, OUT_LSHIFT);

```

3.6.3 riscv_nn_avepool_HWC_s8_any_act

Prototype:

```

int32_t riscv_nn_avepool_HWC_s8_any_act (const int
in_tensor_dim_y, const int in_tensor_dim_x, const int out_tensor_dim_y,
const int out_tensor_dim_x, const int stride_y, const int stride_x, const int
ker_dim_y, const int ker_dim_x, const int pad_y, const int pad_x, const int
act_min, const int act_max, const int in_tensor_ch, int8_t * in_tensor,
int16_t * in_tmp_buf, int8_t * out_tensor)

```

Description

This is an average pooling function for signed 8-bit integer inputs in any x and y dimensions with activation parameters to limit the outputs.

Parameter:

- [in] `in_tensor_dim_y` y dimension of the input tensor
- [in] `in_tensor_dim_x` x dimension of the input tensor
- [in] `out_tensor_dim_y` y dimension of the output tensor
- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `stride_y` convolution stride in the y dimension
- [in] `stride_x` convolution stride in the x dimension

- [in] `ker_dim_y` y dimension of the convolution kernel
- [in] `ker_dim_x` x dimension of the convolution kernel
- [in] `pad_y` padding size in the y dimension
- [in] `pad_x` padding size in the x dimension
- [in] `act_min` minimum value that the output tensor is limited to, and the range is -128 to 127
- [in] `act_max` maximum value that the output tensor is limited to, and the range is -128 to 127
- [in] `in_tensor_ch` number of input tensor channels
- [in] `*in_tensor` pointer of the input tensor
- [in] `*in_tmp_buf` pointer of temporary buffer for the input tensor
- [out] `*out_tensor` pointer of the output tensor

Return Value:

0

3.6.4 `riscv_nn_avepool_HWC_s8_any_act_get_buffer_size`

Prototype:

```
int32_t riscv_nn_avepool_HWC_s8_any_act_get_buffer_size (const
int out_tensor_dim_x, const int in_tensor_ch)
```

Description

This function is used to obtain the required size, in bytes, for the input temporary buffer of `riscv_nn_avepool_HWC_s8_any_act`.

Parameter:

- [in] `out_tensor_dim_x` x dimension of the output tensor
- [in] `in_tensor_ch` number of input tensor channels

Return Value:

This function returns the size required by the temporary buffer.

3.6.5 `riscv_nn_maxpool_HWC_s8`

Prototype:

```
void riscv_nn_maxpool_HWC_s8 (q7_t * in_tensor, const uint16_t
in_tensor_dim, const uint16_t in_tensor_ch, const uint16_t ker_dim, const
uint16_t pad, const uint16_t stride, const uint16_t out_tensor_dim, q7_t *
in_tmp_buf, q7_t * out_tensor)
```

Description

This is an max pooling function for signed 8-bit integer inputs.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_dim` dimension of the input tensor
- [in] `in_tensor_ch` number of input tensor channels
- [in] `ker_dim` dimension of the convolution kernel
- [in] `pad` padding size
- [in] `stride` convolution stride
- [in] `out_tensor_dim` dimension of the output tensor
- [in] `*in_tmp_buf` dummy pointer
- [out] `*out_tensor` pointer of the output tensor

Return Value:

None

Example:

```
#define IN_DIM 32
#define IN_CH 32
#define KER_DIM 3
#define PAD 0
#define STRIDE 2
#define OUT_DIM 15
q7_t in_data[IN_CH * IN_DIM * IN_DIM] = {...};
q7_t out_data[IN_CH * OUT_DIM * OUT_DIM] = {...};
riscv_nn_maxpool_HWC_s8 (in_data, IN_DIM, IN_CH, KER_DIM,
PAD, STRIDE, OUT_DIM, NULL, out_data);
```

3.6.6 riscv_nn_maxpool_HWC_s8_any_act**Prototype:**

```
int32_t riscv_nn_maxpool_HWC_s8_any_act (const uint16_t
in_tensor_dim_y, const uint16_t in_tensor_dim_x, const uint16_t
out_tensor_dim_y, const uint16_t out_tensor_dim_x, const uint16_t
stride_y, const uint16_t stride_x, const uint16_t ker_dim_y, const uint16_t
ker_dim_x, const uint16_t pad_y, const uint16_t pad_x, const int8_t
act_min, const int8_t act_max, const uint16_t in_tensor_ch, int8_t *
```

`in_tensor, int16_t * tmp_buffer, int8_t * out_tensor)`

Description

This is a max pooling function for signed 8-bit integer inputs in any x and y dimensions with activation parameters to limit the outputs.

Parameter:

- `[in] in_tensor_dim_y` y dimension of the input tensor
- `[in] in_tensor_dim_x` x dimension of the input tensor
- `[in] out_tensor_dim_y` y dimension of the output tensor
- `[in] out_tensor_dim_x` x dimension of the output tensor
- `[in] stride_y` convolution stride in the y dimension
- `[in] stride_x` convolution stride in the x dimension
- `[in] ker_dim_y` y dimension of the convolution kernel
- `[in] ker_dim_x` x dimension of the convolution kernel
- `[in] pad_y` padding size in the y dimension
- `[in] pad_x` padding size in the x dimension
- `[in] act_min` minimum value that the output tensor is limited to, and the range is -128 to 127
- `[in] act_max` maximum value that the output tensor is limited to, and the range is -128 to 127
- `[in] in_tensor_ch` number of input tensor channels
- `[in] *in_tensor` pointer of the input tensor
- `[in] *tmp_buf` dummy pointer
- `[out] *out_tensor` pointer of the output tensor

Return Value:

0

3.7 Softmax Functions

The softmax functions are exponential functions with base 2.

3.7.1 riscv_nn_softmax_s8_fast

Prototype:

```
void riscv_nn_softmax_s8_fast (const q7_t * in_vec, const uint16_t
size, q7_t * out_vec)
```

Description

This is a softmax function for signed 8-bit integer input vectors.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] size` number of elements in the input vector
- `[out] *out_vec` pointer of the output vector

Return Value:

None

Example:

```
#define LENGTH 10
q7_t in_data[LENGTH] = {...};
q7_t out_data[LENGTH];
riscv_nn_softmax_s8_fast (in_data, LENGTH, out_data);
```

3.7.2 riscv_nn_softmax_s16_fast**Prototype:**

```
void riscv_nn_softmax_s16_fast (const q15_t * in_vec, const uint16_t
size, q15_t * out_vec)
```

Description

This is a softmax function for signed 16-bit integer input vectors.

Parameter:

- `[in] *in_vec` pointer of the input vector
- `[in] size` number of elements in the input vector
- `[out] *out_vec` pointer of the output vector

Return Value:

None

3.7.3 riscv_nn_softmax_s8_hp**Prototype:**

```
void riscv_nn_softmax_s8_hp (const int8_t * in_tensor, const int32_t
in_tensor_row, const int32_t in_tensor_col, const int32_t scale, const
int32_t lshift, const int32_t diff_min, int8_t * out_tensor)
```

Description

This is a softmax function for signed 8-bit integer input tensor with

high precision algorithm.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_row` number of rows in the input tensor
- [in] `in_tensor_col` number of columns in the input tensor
- [in] `scale` scaling value for input quantization
- [in] `lshift` left shift amount for the input quantization
- [in] `diff_min` minimum threshold to perform the quantized exponential operation. The difference can be obtained by subtracting the input from the maximum in row
- [out] `*out_tensor` pointer of the output tensor

Return Value:

None

3.7.4 `riscv_nn_softmax_u8_hp`

Prototype:

```
void riscv_nn_softmax_u8_hp (const uint8_t * in_tensor, const int32_t  
in_tensor_row, const int32_t in_tensor_col, const int32_t scale, const  
int32_t lshift, const int32_t diff_min, uint8_t * out_tensor)
```

Description

This is a softmax function for unsigned 8-bit integer input tensor with high precision algorithm.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [in] `in_tensor_row` number of rows in the input tensor
- [in] `in_tensor_col` number of columns in the input tensor
- [in] `scale` scaling value for input quantization
- [in] `lshift` left shift amount for the input quantization
- [in] `diff_min` minimum threshold to perform the quantized exponential operation. The difference can be obtained by subtracting the input from the maximum in row
- [out] `*out_tensor` pointer of the output tensor

Return Value:

None

3.7.5 riscv_nn_softmax_f16

Prototype:

```
int32_t riscv_nn_softmax_f16 (const float16_t * in_vec, const uint32_t  
size, float16_t * out_vec)
```

Description

This is a softmax function for half-precision floating-point input vectors.

Parameter:

- [in] **in_vec* pointer of the input vector
- [in] *size* number of elements in the input vector
- [out] **out_vec* pointer of the output vector

Return Value:

0

3.8 Utils Functions

Utils functions are miscellaneous auxiliary tools.

3.8.1 riscv_nn_exp_f16

Prototype:

```
int32_t riscv_nn_exp_f16 (const float16_t * in_vec, const uint32_t  
size, float16_t * out_vec)
```

Description

This function calculates the base-e exponential values of half-precision floating-point inputs.

Parameter:

- [in] **in_vec* pointer of the input vector
- [in] *size* number of elements in the input vector
- [out] **out_vec* pointer of the output vector

Return Value:

0

3.8.2 riscv_nn_reshape_s8

Prototype:

```
void riscv_nn_reshape_s8 (const int8_t * in_tensor, int8_t *  
out_tensor, const uint32_t size)
```

Description

This function turns the input tensor into another tensor with the same data but in a different shape.

Parameter:

- [in] `*in_tensor` pointer of the input tensor
- [out] `*out_tensor` pointer of the output tensor
- [in] `size` size, in bytes, of total input tensors

Return Value:

None

Example:

```
#define SIZE 1024  
int8_t in_tensor[SIZE] = {...};  
int8_t out_tensor[SIZE];  
riscv_nn_reshape_s8 (in_tensor, out_tensor, SIZE);
```

3.8.3 riscv_nn_top_k_s8

Prototype:

```
int32_t riscv_nn_top_k_s8 (q7_t * in_vec, uint32_t size, uint32_t k,  
q7_t * val, uint32_t * idx)
```

Description

This function finds the k largest values and their indices from the signed 8-bit integer input vector.

Parameter:

- [in] `*in_vec` pointer of the input vector
- [in] `size` number of elements in the input vector
- [in] `k` the number of the largest values to be searched
- [out] `*val` the k largest values in the input vector
- [out] `*idx` the indices of the k largest values in the input vector

Return Value:

0

Note!

The k largest values will be sorted from the largest to the smallest and stored in the "val" output vector. If multiple elements share the same value, those with smaller indices will have higher priority for the selection.

3.8.4 riscv_nn_top_k_f16

Prototype:

```
int32_t riscv_nn_top_k_f16 (float16_t * in_vec, uint32_t size, uint32_t k, float16_t * val, uint32_t * idx)
```

Description

This function finds the k largest values and their indices from the half-precision floating-point input vector.

Parameter:

- [in] **in_vec* pointer of the input vector
- [in] *size* number of elements in the input vector
- [in] *k* the number of the largest values to be searched
- [out] **val* the k largest values in the input vector
- [out] **idx* the indices of the k largest values in the input vector

Return Value:

0

Note!

The k largest values will be sorted from the largest to the smallest and stored in the "val" output vector. If multiple elements share the same value, those with smaller indices will have higher priority for the selection.

4 Application Program

4.1 NN CIFAR-10

4.1.1 Program Description

This program includes functions for convolution, ReLU, max pooling, fully connected etc., demonstrating accelerated computations on a RISC-V processor. It uses a trained CIFAR-10 neural network model for inference and recognition of 10 targets (0-9), representing airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

The inference results of this program show that the target "8" has the highest score of 127, while the scores for other targets are 0. Therefore, the input image used for testing is most likely a ship.

The output information of this program includes instruction count, cycle count, inference scores for the testing image, and accuracy verification of the results.

4.1.2 Application Program

RiscV_AE350_SOC provides NN CIFAR-10 application programming:
...\ref_design\MCU_RefDesign\ae350_nn_cifar10

4.1.3 Program Running

The results of running NN CIFAR-10 application program are shown below.

It is a Neural Network Model with the CIFAR-10 dataset demo.
Start execution
Start...

```

after init the checksum of img_buffer2 = 00000000093b43f6
after riscv_nn_conv_HWC_s8_s8_s8_RGB_sft_bias_fast the checksum
of img_buffer1 = ffffffff8e2c90
after riscv_nn_relu_s8 the checksum of img_buffer1 =
000000000023b2f0
after riscv_nn_maxpool_HWC_s8 the checksum of img_buffer2 =
00000000001c94ee
after riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast the checksum of
img_buffer1 = ffffffff8b1ed7e
after riscv_nn_relu_s8 the checksum of img_buffer1 =
000000000007f614
after riscv_nn_maxpool_HWC_s8 the checksum of img_buffer2 =
0000000000076bca
after riscv_nn_conv_HWC_s8_s8_s8_sft_bias_fast the checksum of
img_buffer1 = ffffffff8a57b4
after riscv_nn_relu_s8 the checksum of img_buffer1 =
0000000000012fc8
after riscv_nn_maxpool_HWC_s8 the checksum of img_buffer2 =
0000000000018662
after riscv_nn_fc_s8_s8_s8_sft_bias_fast the checksum of output_data
= 000000000000464c
after riscv_nn_softmax_s8_fast the checksum of output_data =
000000000001f4208
Inst: 52064421, Cycle: 374343076
0:airplane      0
1:automobile    0
2:bird          0
3:cat           0
4:deer          0
5:dog           0
6:frog          0
7:horse         0
8:ship         127
9:truck         0
>>>> Accuracy checking 'ship' PASS. <<<<

```

4.2 NN MobileNet-V1 INT8

4.2.1 Program Description

This program demonstrates the NN MobileNet model. MobileNet is a model that uses depth-wise separable convolutions to build lightweight and efficient deep neural networks. It is primarily used in the mobile and embedded vision domains for applications such as object detection and image classification.

The program represents a quantized MobileNet model with INT8. Output information includes instruction count, cycle count, inference scores for the testing image, and accuracy verification of the results.

4.2.2 Application Program

RiscV_AE350_SOC provides NN MobileNet-V1 INT8 application programming:

```
...\ref_design\MCU_RefDesign\ae350_nn_mobilenet_v1_int8
```

4.2.3 Program Running

The results of running NN MobileNet-V1 INT8 application program are shown below.

```
It's a Neural Network with MobileNet-V1 INT8 model demo.
Start executing...
Start...
after init the checksum of image_data = 00000001233bdcf2
after riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast the
checksum of buffer1 = ffffffff9e93e9ee
after riscv_nn_relu_s8 the checksum of buffer1 = 0000001d0d2d385a
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = fffffffe68949866
after riscv_nn_relu_s8 the checksum of buffer2 = 00000009f1729db6
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 00000002a28c0594
after riscv_nn_relu_s8 the checksum of buffer1 = 000000064e9bde6c
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 0000000143325e50
```

after riscv_nn_relu_s8 the checksum of buffer2 = 000000039f7f55c0
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 00000003d23f3a6a
after riscv_nn_relu_s8 the checksum of buffer1 = 000000085ec8093a
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 000000005f0a4bf0
after riscv_nn_relu_s8 the checksum of buffer2 = 0000000316896b76
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffff66efeafc
after riscv_nn_relu_s8 the checksum of buffer1 = 000000012361dbc4
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 00000000a16ee37e
after riscv_nn_relu_s8 the checksum of buffer2 = 00000000b4775278
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 000000018a0613ca
after riscv_nn_relu_s8 the checksum of buffer1 = 000000019a0099e6
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 00000000114b864a
after riscv_nn_relu_s8 the checksum of buffer2 = 000000009287b20c
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffff6e7220
after riscv_nn_relu_s8 the checksum of buffer1 = 000000004624aba0
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 000000005b0da5fa
after riscv_nn_relu_s8 the checksum of buffer2 = 0000000068700136
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 00000000871ff516
after riscv_nn_relu_s8 the checksum of buffer1 = 000000009229946c
after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = ffffffffcd3e2ca2
after riscv_nn_relu_s8 the checksum of buffer2 = 000000003fd73c5c
after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 000000000b876592
after riscv_nn_relu_s8 the checksum of buffer1 = 000000001e2aa242

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = ffffffff797f224

after riscv_nn_relu_s8 the checksum of buffer2 = 000000001358ebb4

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 0000000001f8766a

after riscv_nn_relu_s8 the checksum of buffer1 = 00000000189c9a12

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 0000000007f8becc

after riscv_nn_relu_s8 the checksum of buffer2 = 0000000050d2bdd6

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffff3b226

after riscv_nn_relu_s8 the checksum of buffer1 = 000000000a8bd970

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 00000000114adb1e

after riscv_nn_relu_s8 the checksum of buffer2 = 000000003179d8fc

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffff846a78

after riscv_nn_relu_s8 the checksum of buffer1 = 000000000b6d1b50

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 000000000b208e9a

after riscv_nn_relu_s8 the checksum of buffer2 = 00000000202d3a44

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = 00000000056b9676

after riscv_nn_relu_s8 the checksum of buffer1 = 0000000014d57b56

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 000000000a4c525c

after riscv_nn_relu_s8 the checksum of buffer2 = 000000000e55a490

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffff4d4486a

after riscv_nn_relu_s8 the checksum of buffer1 = 0000000002f7646e

after riscv_nn_conv_dw_HWC_s8_s8_s8_sym_bias the checksum of
buffer2 = 000000000cfd3304

after riscv_nn_relu_s8 the checksum of buffer2 = 00000000266a863c

after riscv_nn_conv_1x1_HWC_s8_s8_s8_sym_bias_fast_any the
checksum of buffer1 = ffffffffed98ba3a

```

after riscv_nn_relu_s8 the checksum of buffer1 = 0000000000bb5982
after riscv_nn_avepool_HWC_s8 the checksum of buffer2 =
0000000000000001
after riscv_nn_fc_s8_s8_s8_sym_bias the checksum of out_data =
00000000000baa8e
after riscv_nn_softmax_s8_fast the checksum of out_data =
00000000001f45e6
Inst: 90956449, Cycle: 119727866
>>>>> Accuracy checking PASS. <<<<<

```

4.3 NN TinyYOLO-V3

4.3.1 Program Description

This program demonstrates the NN TinyYOLO-V3 model. TinyYOLO is a variant of the real-time object detection system "You Only Look Once" (YOLO), featuring a smaller model size for constrained environments.

The TinyYOLO-V3 in this program is a lightweight version of YOLO with a simplified backbone network and two-level feature map detection, primarily designed for resource-limited devices such as embedded systems, mobile devices, or IoT devices.

Output information from the program includes instruction count, cycle count, and accuracy verification of the results.

4.3.2 Application Program

RiscV_AE350_SOC provides NN TinyYOLO-V3 application programming:

```
...ref_design\MCU_RefDesign\ae350_nn_tinyyolo_v3
```

4.3.3 Program Running

The results of running NN TinyYOLO-V3 application program are shown below.

```

It's a Neural Network with TinyYOLO v3 model demo.
Start executing model...
Start...
after init the checksum of image_data = 00000036bdd17bbc

```

after riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast the checksum of buffer1 = fffffff2efaf75b4

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 00000097c470d19c

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 000000034f10415c

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of buffer1 = 0000000593c74a7c

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 00000113b0f644ba

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 0000000362cb292c

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of buffer1 = fffffffddb234434

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 000000679282c680

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 00000000fca7413c

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of buffer1 = fffffffeea8c97a0

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 00000033b51118da

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 000000003fcde4b0

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of buffer1 = ffffffffb1c72628

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 0000000a6634634c

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 0000000007f79719

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of buffer1 = ffffffffac79e31a

after riscv_nn_leaky_relu_s8 the checksum of buffer1 = 0000000cf280f62a

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 = 000000000f8e086f

```

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = 00000000469cc836

after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
0000001a1a8ff9aa

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer2 = ffffffff7c14f22

after riscv_nn_leaky_relu_s8 the checksum of buffer2 =
000000065365f146

=====run route...

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer3 = ffffffff7010c7a

after riscv_nn_leaky_relu_s8 the checksum of buffer3 =
00000003308f3bfe

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer4 = ffffffff2d3a0b4a

after riscv_nn_leaky_relu_s8 the checksum of buffer4 =
00000021c13e7396

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer3 = 000000151f27449e

=====run route end...

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = ffffffff220b478

after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
0000000d1339f80c

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer2 = 000000064cfe76f5

Inst: -174936110, Cycle: 1589518486

Done

```

4.4 NN TinyYOLO-V1

4.4.1 Program Description

This program demonstrates the NN TinyYOLO-V1 model. TinyYOLO is a variant of the real-time object detection system "You Only Look Once" (YOLO) with a smaller model size suitable for constrained environments.

The TinyYOLO-V1 model in this program is quantized to INT8 using a scheme involving symmetrically activated data (i.e., enforcing zero point to

be zero).

Output information from the program includes instruction count, cycle count, and accuracy verification of the results.

4.4.2 Application Program

RiscV_AE350_SOC provides NN TinyYOLO-V1 application programming:

...\ref_design\MCU_RefDesign\ae350_nn_tinyyolo_v1

4.4.3 Program Running

The results of running NN TinyYOLO-V1 application program are shown below.

```
It's a Neural Network with TinyYOLO v1 model demo.
Start executing model...
Start...
after init the checksum of image_data = 00000036bdd17bbc
after riscv_nn_conv_HWC_s8_s8_s8_RGB_sym_bias_fast the
checksum of buffer1 = fffffffe476b89c4
after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
00000001473069d6
after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
000000000b3008e4
after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffff21105a370
after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
0000000075d18612
after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
00000000017914b2
after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffffbc5f86c94
after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
000000005b321f12
after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
0000000000d8461e
after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffffdf2c7fe70
```

```
after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
000000002003068c

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
00000000001a0532

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffffebc0181aa

after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
0000000007d995d0

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
00000000000320fa

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffff1af40ea2

after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
0000000002090daa

after riscv_nn_maxpool_HWC_s8 the checksum of buffer2 =
0000000000005da3

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer1 = fffffffb491053a

after riscv_nn_leaky_relu_s8 the checksum of buffer1 =
000000000268bece

after riscv_nn_conv_HWC_s8_s8_s8_sym_bias_fast the checksum of
buffer2 = fffffffecc0bd14

after riscv_nn_leaky_relu_s8 the checksum of buffer2 =
0000000000af9b74

after riscv_nn_fc_s8_s8_s8_sym_bias the checksum of out_data =
0000000001a5f0a0

Inst: -1345112762, Cycle: 1042775620

Done
```

