NNTurbo API 指南

文档版本号：v1.0

发布日期：2020-1-10

商标声明


、全志和其他全志商标均为珠海全志科技股份有限公司的商标。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受全志公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，全志公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保

前言

概述

本文档为基于 NNTurbo 开发算法模型的工程师提供开发指导。

产品版本

与本文档对应的产品版本。

产品名称	产品版本
V833	NNTurbo V1.0

读者对象

本文档（本指南）主要适用于以下工程师：

使用 NNTurbo 进行开发的软件/算法开发工程师

名词解释

API——应用程序接口,全称：Application Programming Interface

SDK——软件开发包，全称：Software Development Kit

NNTurbo——AW 中间件总称

修订记录

版本号	修订日期	修订内容
NNTurbo V1.0	2020-1-10	第一次发布。

目录

1. NNTurbo 介绍

NNTurbo 是 AW AI 套件的总称，其中分为两大模块，离线工具和在线 Runtime，二者配合使用。

2. NNTurbo 概述

2.1. 数据格式

NNTurbo 目前只支持 INT8（有符号 8 位）运算，主要是针对权重，特征图以及输入图像。

2.2. 算子支持

算子	描述
conv(卷积)	卷积算子，bias 建议按通道偏置的模式，按 layer 和 point 也可以。
pooling(池化)	池化算子，池化核 <= 7x7。
eltwise(元素对位运算)	支持加法，乘法，乘加运算。
lrn(局部响应归一化)	暂不使用。
inner_product(内积运算)	全连接算子。
activation(激活运算)	支持 relu，prelu。
bn(批归一化)	采用通道模式进行批归一化(Batch Normalization)操作。
conv_act_pool (卷积、激活、池化复合运算)	卷积，激活，池化的融合算子，数据会在模块内部 bypass，实现减少带宽的目的，限制与单个算子的限制相同。
conv_act (卷积、激活复合运算)	卷积，激活融合算子，数据会在模块内部 bypass，实现减少带宽的目的，限制与单个算子的限制相同。
conv_act_eltwise (卷积、激活、元素对位复合运算)	卷积，激活，对位运算算子，数据会在模块内部 bypass，实现减少带宽的目的，限制与单个算子的限制相同。（此算子主要用于残差网络运算。）
conv_bn_pool （卷积、bn、池化复合运算）	卷积，batch normalization，池化融合算子，数据会在模块内部 bypass，实现减少带宽的目的，限制与单个算子的限制相同。
conv_act_bn （卷积、激活、bn 复合运算）	注意 prelu 与 bn 不能同时使用。
conv_act_bn_pool （卷积、激活、bn、池化复合运算）	注意 prelu 与 bn 不能同时使用。

2.3. 框架支持

当前只支持 TensorFlow 的模型，其它框架的模型需要转换成 TensorFlow 模型。

2.4. 注意事项

受 IPU 单元 Buffer 大小约束，参与运算的特征图与权重占内存的总大小不能超过 256KB,若超出，则需保证特征图+8 通道（8 kernels）权重值大小超过 256KB，工具会对权重按 8 通道（8 kernels）切分，以上按 int8 计算。

卷积核的通道数为 32 的倍数时，效率最高；其它也能运行。

3. NNTurbo API 描述

3.1. 模块接口

3.1.1. hw_init

【描述】

硬件模块初始化函数。

【语法】

```
void hw_init();
```

【参数】

参数名称	描述	输入/输出
/	/	/

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.1.2. hw_reset

【描述】

硬件模块重置函数，每启动一次，都需要执行一次。

【语法】

```
void hw_reset();
```

【参数】

参数名称	描述	输入/输出
/	/	/

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.1.3. hw_deinit

【描述】

硬件模块释放函数。

【语法】

```
void hw_deinit();
```

【参数】

参数名称	描述	输入/输出
/	/	/

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.1.4. readl_sig

【描述】

获取算子运算是否完毕的数值信号。

【语法】

```
unsigned int readl_sig()
```

【参数】

参数名称	描述	输入/输出
/	/	/

【返回值】

参数类型	描述	返回值
unsigned int	表达运算完毕的数值信号。	请参见附录“运算完毕信号”

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.2. NN 接口

3.2.1. aw_ai_conv_program

【描述】

针对卷积算子的操作函数，此函数为阻塞方式。

【语法】

```
int aw_ai_conv_program(struct aw_ai_conv_op_desc conv_op_desc);
```

【参数】

参数名称	描述	输入/输出
conv_op_desc	算子的描述结构体，具体参数描述请参考结构体部分。	输入

【返回值】

参数类型	描述	返回值
int	标识当前卷积算子操作函数在网络中的索引。	/

【参考头文件】

```
ipu_runtime.h
```

【例程】

【备注】

3.2.2. aw_ai_pool_program

【描述】

针对池化算子的操作函数，此函数为阻塞方式。

【语法】

```
int aw_ai_pool_program(struct aw_ai_pool_op_desc pool_op_desc);
```

【参数】

参数名称	描述	输入/输出
pool_op_desc	算子的描述结构体，具体参数描述请参考结构体部分	输入

【返回值】

参数类型	描述	返回值
int	标识当前池化算子操作函数在网络中的索引。	/

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.3. aw_ai_conv_pool_program

【描述】

卷积与池化融合执行的操作函数，此函数为阻塞形式。

【语法】

```
int aw_ai_conv_pool_program(struct aw_ai_conv_op_desc conv_op_desc,
                            struct aw_ai_pool_op_desc pool_op_desc);
```

【参数】

参数名称	描述	输入/输出
conv_op_desc	算子的描述结构体，具体参数描述请参考结构体部分。	输入
pool_op_desc	算子的描述结构体，具体参数描述请参考结构体部分。	输入

【返回值】

参数类型	描述	返回值
int	标识当前卷积与池化融合执行的操作函数在网络中的索引。	/

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.4. aw_ai_element_wise_program

【描述】

针对逐元素计算的操作函数，实现矩阵间的乘法、加法及乘加操作，此函数为阻塞方式。

【语法】

```
int aw_ai_element_wise_program(struct aw_ai_element_wise_op_desc element_op_desc);
```

【参数】

参数名称	描述	输入/输出
element_op_desc	算子的描述结构体，具体参数描述请参考结构体部分。	输入

【返回值】

参数类型	描述	返回值
int	标识当前逐元素计算的操作函数在网络中的索引。	/

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.5. aw_ai_op_completion

【描述】

判断当前算子的操作函数是否执行结束，并释放缓存中相关的资源。

【语法】

```
int aw_ai_op_completion(unsigned int idx);
```

【参数】

参数名称	描述	输入/输出
idx	算子操作函数对应在网络中的索引。	输入

【返回值】

参数类型	描述	返回值
int	操作是否成功的标志位。	完成返回 1。

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.6. aw_ai_cvt_data_hwc2ipu

【描述】

数据转换函数，将 hwc 格式的数据，转换成硬件模块可读的数据。

【语法】

```
int aw_ai_cvt_data_hwc2ipu(void* ptr_src, int height, int width, int channels,
                           void* ptr_dst);
```

【参数】

参数名称	描述	输入/输出
ptr_src	源地址，虚拟地址。	输入
height	高度。	输入
width	宽度。	输入
channels	通道数。	输入
ptr_dst	目标地址，虚拟地址。	输入

【返回值】

参数类型	描述	返回值
int	操作是否成功的标志位。	完成返回 1。

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.7. aw_ai_cvt_data_ipu2hwc

【描述】

数据转换函数，将硬件模块的数据格式，转换成 hwc 格式。

【语法】

```
int aw_ai_cvt_data_ipu2hwc(void* ptr_src, int height, int width, int channels,
                           void* ptr_dst);
```

【参数】

参数名称	描述	输入/输出
ptr_src	源地址，虚拟地址。	输入
height	高度。	输入
width	宽度。	输入
channels	通道数。	输入
ptr_dst	目标地址，虚拟地址。	输入

【返回值】

参数类型	描述	返回值
int	操作是否成功的标志位。	完成返回 1。

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.2.8. run_ops

【描述】

针对多算子运行的操作函数，也即统一的算子运行接口。

【语法】

```
void run_ops(struct ipu_ops_params* p_ipu_ops, int first_layer);
```

【参数】

参数名称	描述	输入/输出
p_ipu_ops	算子描述指针。	输入
first_layer	是否是图像层（第一层）。	输入

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

ipu_runtime.h

【例程】

【备注】

3.3. 内存接口

3.3.1. dma_mem_alloc

【描述】

分配连续的物理内存。

【语法】

```
void dma_mem_alloc(unsigned int size, void** p_vaddr, void** p_paddr);
```

【参数】

参数名称	描述	输入/输出
size	需分配内存的大小，以字节为单位。	输入
p_vaddr	分配内存的虚拟地址。	输入/输出
p_paddr	分配内存的物理地址。	输入/输出

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.3.2. dma_mem_free

【描述】

释放物理内存的操作函数。

【语法】

```
void dma_mem_free(void* vaddr);
```

【参数】

参数名称	描述	输入/输出
vaddr	分配内存的虚拟地址。	输入

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

hw_adaptor.h

【例程】

【备注】

3.3.3. mem_alloc

【描述】

分配连续的常规虚拟内存。

【语法】

```
void mem_alloc(P_MEM_CTRL ptr_info, unsigned int size);
```

【参数】

参数名称	描述	输入/输出
ptr_info	分配内存的结构体指针	输入/输出
size	分配内存的大小，byte	输入

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

mem_ctrl.h

【例程】

【备注】

3.3.4. mem_free

【描述】

释放虚拟内存的操作函数。

【语法】

```
void mem_free(P_MEM_CTRL ptr_info);
```

【参数】

参数名称	描述	输入/输出
ptr_info	分配内存的结构体指针。	输入

【返回值】

参数类型	描述	返回值
/	/	/

【参考头文件】

mem_ctrl.h

【例程】

【备注】

4. 结构体说明

4.1.aw_ai_conv_op_desc

【说明】

卷积函数使用的数据结构体。

【定义】

```
struct aw_ai_conv_op_desc {
    /* Performance parameters */
    uint8_t first_layer;
    uint8_t data_format;
    /* The input cube dimension for CSC */
    uint32_t input_data_addr;
    uint16_t input_width;
    uint16_t input_height;
    uint16_t input_channel;

    uint32_t kernel_data_addr;
    uint16_t kernel_width;
    uint16_t kernel_height;
    uint16_t kernel_channel;
    uint16_t kernel_num;

    //0->disable,1->layer,2->pre channel,3->pre elem
    uint8_t bias_mode;
    uint32_t bias_data_addr;

    uint8_t conv_stride_x;
    uint8_t conv_stride_y;
    uint8_t pad_x_left;
    uint8_t pad_x_right;
    uint8_t pad_y_top;
    uint8_t pad_y_bottom;
    uint8_t dilation_x;
    uint8_t dilation_y;
```

```
uint8_t reserved2[2];

uint8_t mean_enable;
int16_t mean_ry; /* mean value for red in RGB or Y in YUV */
int16_t mean_gu; /* mean value for green in RGB or U in YUV */
int16_t mean_bv; /* mean value for blue in RGB or V in YUV */
int16_t mean_ax;
int16_t pad_val;

/*act*/
//0->disable,1->relu,2->prelu
uint8_t act_enbale;
uint32_t act_data_addr;

/*bn*/
//0->disable,1->enable,BN 和 prelu 不能同时用
uint8_t bn_enbale;
uint32_t bn_data_addr;

/*eltwise*/
//0->disable,1->enable,BN 和 prelu 不能同时用
uint8_t eltwise_enbale;   uint32_t eltwise_data_addr;
/* output converter parameters, support truncate only */
uint8_t  cvt_enable;
int16_t  scale;
uint8_t  truncate;
int32_t  offset;

uint32_t output_data_addr;
uint16_t output_width;
uint16_t output_height;
uint16_t output_channel;

uint32_t reverse1;
uint32_t reverse2;
```

};

【变量】

变量	含义	类型	取值范围
first_layer	是否是第一层： 0：不是。 1：是。	uint8_t	/
data_format	数据格式，具体可参考附录。	uint8_t	/
input_data_addr	输入数据的物理地址。	uint32_t	/
input_width	输入数据的宽度。	uint16_t	/
input_height	输入数据的高度。	uint16_t	/
input_channel	输入数据的通道数。	uint16_t	/
kernel_data_addr	卷积核权重数据的物理地址。	uint32_t	/
kernel_width	卷积核宽度。	uint16_t	/
kernel_height	卷积核高度。	uint16_t	/
kernel_channel	卷积核通道数。	uint16_t	/
kernel_num	卷积核个数。	uint16_t	/
bias_mode	偏置模式。	uint8_t	0->disable, 1->layer, 2->pre channel, 3->pre elem
bias_data_addr	偏置数据地址。	uint32_t	/
conv_stride_x	卷积滑动步长，水平方向。	uint8_t	1~3
conv_stride_y	卷积滑动步长，垂直方向。	uint8_t	1~3
pad_x_left	pad 个数，默认 pad 值是 0。	uint8_t	0~1
pad_x_right	pad 个数，默认 pad 值是 0。	uint8_t	0~1
pad_y_top	pad 个数，默认 pad 值是 0。	uint8_t	0~1
pad_y_bottom	pad 个数，默认 pad 值是 0。	uint8_t	0~1
dilation_x	dilation 卷积，水平方向像素个数。	uint8_t	/
dilation_y	dilation 卷积，垂直方向像素个数。	uint8_t	/
mean_enable	均值偏差使能开关。	uint8_t	目前默认 0
mean_ry	通道 1 均值。	int16_t	目前默认 0
mean_gu	通道 2 均值。	int16_t	目前默认 0
mean_bv	通道 3 均值。	int16_t	目前默认 0

mean_ax	通道 4 均值。	int16_t	目前默认 0
pad_val	pad 的数值。	int16_t	目前默认 0
act_enbale	激活使能。	uint8_t	0->disable 1->relu 2->prelu
act_data_addr	激活参数物理地址。	uint32_t	只对 prelu 有效
bn_enbale	bn 使能。	uint8_t	0->disable 1->enable BN 和 prelu 不能同时用
bn_data_addr	bn 参数物理地址。	uint32_t	/
eltwise_enbale	元素累加使能。	uint8_t	0->disable 1->enable
eltwise_data_addr	元素累加源数据物理地址。	uint32_t	/
cvt_enable	输出数值转换使能。	uint8_t	0->disable 1->enable
scale	输出数值 scale 因子。	int16_t	/
truncate	输出数值饱和值（右移 bit 数）。	uint8_t	/
offset	输出数值偏置值。	int32_t	/
output_data_addr	输出数据物理地址。	uint32_t	/
output_width	输出数据宽度。	uint16_t	/
output_height	输出数据高度。	uint16_t	/
output_channel	输出数据通道数。	uint16_t	/
reverse1	保留参数。	uint32_t	默认值 0
reverse2	保留参数。	uint32_t	默认值 0

【参考头文件】

ipu_runtime.h

【备注】

4.2.aw_ai_element_wise_op_desc

【说明】

逐元素操作函数使用的结构体。

【定义】

```
struct aw_ai_element_wise_op_desc {
    /* Performance parameters */
    uint8_t data_format;
    uint8_t operation_type; //0-mul,1-add,2-mul+add

    uint32_t input1_data_addr;
    uint16_t input1_width;
    uint16_t input1_height;
    uint16_t input1_channel;

    uint32_t input2_data_addr;
    uint16_t input2_width;
    uint16_t input2_height;
    uint16_t input2_channel;

    uint32_t input3_data_addr;
    uint16_t input3_width;
    uint16_t input3_height;
    uint16_t input3_channel;

    int32_t add_operand;
    int32_t mul_operand;

    /*默认输入输出都是 INT8，所以输出要转换量化。
    offset -> scale -> truncate(右移截断)*/
    int16_t scale;
    uint8_t cvt_enable;
    uint8_t  truncate;
    int32_t  offset;

    uint32_t output_data_addr;
    uint16_t output_width;
    uint16_t output_height;
    uint16_t output_channel;
```

};

【变量】

变量	含义	类型	取值范围
data_format	数据格式，请参照数据格式说明。	uint8_t	/
operation_type	操作类型。	uint8_t	0-Mul 1-Add 2-mul+add
input1_data_addr	输入数据 1 的物理地址。	uint32_t	/
input1_width	输入数据 1 的宽度。	uint16_t	/
input1_height	输入数据 1 的高度。	uint16_t	/
input1_channel	输入数据 1 的通道数。	uint16_t	/
input2_data_addr	输入数据 2 的物理地址。	uint32_t	/
input2_width	输入数据 2 的宽度。	uint16_t	/
input2_height	输入数据 2 的高度。	uint16_t	/
input2_channel	输入数据 2 的通道数。	uint16_t	/
input3_data_addr	输入数据 3 的物理地址。	uint32_t	/
input3_width	输入数据 3 的宽度。	uint16_t	/
input3_height	输入数据 3 的高度。	uint16_t	/
input3_channel	输入数据 3 的通道数。	uint16_t	/
add_operand;	加法操作数。	int32_t	/
mul_operand	乘法操作数。	int32_t	/
cvt_enable	输出数值转换使能。	uint8_t	0->disable 1->enable 默认输入输出都是 INT8，所以输出要转换量化。offset -> scale -> truncate(右移截断)
scale	输出数值 scale 因子。	int16_t	/
truncate	输出数值饱和值（右移 bit 数）。	uint8_t	/
offset	输出数值偏置值。	int32_t	/
output_data_addr	输出数据物理地址。	uint32_t	/
output_width	输出数据宽度。	uint16_t	/

output_height	输出数据高度。	uint16_t	/
output_channel	输出数据通道数。	uint16_t	/

【参考头文件】

ipu_runtime.h

【备注】

4.3.aw_ai_pool_op_desc

【说明】

Pool 操作结构体

【定义】

```
#define POOL_MODE_AVG      0
#define POOL_MODE_MAX      1
#define POOL_MODE_MIN      2

struct aw_ai_pool_op_desc {
    uint32_t input_addr;
    uint16_t input_width;
    uint16_t input_height;
    uint16_t input_channel;

    uint32_t output_addr;
    uint16_t output_width;
    uint16_t output_height;
    uint16_t output_channel;

    /* Algorithm parameters */
    uint8_t  pool_mode;   /* ipu_pool_mode */
    uint8_t  pool_width;  /* ipu_pool_width */
    uint8_t  pool_height; /* ipu_pool_height */

    uint8_t  stride_x;
    uint8_t  stride_y;
```

```
    uint8_t  pad_num;
    uint8_t  pad_left;
    uint8_t  pad_right;
    uint8_t  pad_top;
    uint8_t  pad_bottom;

    uint32_t reverse1;
    uint32_t reverse2;
};
```

【变量】

变量	含义	类型	取值范围
input_addr	输入数据的物理地址。	uint32_t	/
input_width	输入数据的宽度。	uint16_t	/
input_height	输入数据的高度。	uint16_t	/
input_channel	输入数据的通道数。	uint16_t	/
output_addr	输出数据的物理地址。	uint32_t	/
output_width	输出数据的宽度。	uint16_t	/
output_height	输出数据的高度。	uint16_t	/
output_channel	输出数据的通道数。	uint16_t	/
pool_mode	pool 的类型，请参考相关头文件。	uint8_t	/
pool_width	pool 核的宽度。	uint8_t	/
pool_height	pool 核的高度。	uint8_t	/
stride_x	水平方向滑动步长。	uint8_t	/
stride_y	垂直方向滑动步长。	uint8_t	/
pad_num	pad 数量，暂时未使用。	uint8_t	/
pad_left	左边 pad 的数量。	uint8_t	默认 pad 为 0。
pad_right	右边 pad 的数量。	uint8_t	默认 pad 为 0。
pad_top	上边 pad 的数量。	uint8_t	默认 pad 为 0。
pad_bottom	下边 pad 的数量。	uint8_t	默认 pad 为 0。
reverse1	保留参数。	uint32_t	/
reverse2	保留参数。	uint32_t	/

【参考头文件】

ipu_runtime.h

【备注】

4.4.ipu_ops_params

【说明】

卷积，池化，激活，融合结构体

【定义】

```
struct ipu_ops_params
{
    unsigned int dev_type;
    unsigned int op_type;
    unsigned int blob_id;
    unsigned int cvt_scale;
    unsigned int bits;

    int conv_input_shape_n;
    int conv_input_shape_h;
    int conv_input_shape_w;
    int conv_input_shape_c;
    int conv_kernel_shape_n;
    int conv_kernel_shape_h;
    int conv_kernel_shape_w;
    int conv_kernel_shape_c;
    int conv_stride_x;
    int conv_stride_y;
    int conv_pad;
    int conv_dilation;
    int conv_bn_enbale;
    int conv_bias_mode;
    int conv_act_mode;
    int conv_eltwise_enbale;
    int conv_output_shape_n;
    int conv_output_shape_h;
```

```
    int conv_output_shape_w;
    int conv_output_shape_c;
    int pool_input_shape_n;
    int pool_input_shape_h;
    int pool_input_shape_w;
    int pool_input_shape_c;
    int pool_kernel;
    int pool_pad;
    int pool_stride;
    int pool_output_shape_n;
    int pool_output_shape_h;
    int pool_output_shape_w;
    int pool_output_shape_c;

    int in_addr;
    int weight_addr;
    int bias_addr;
    int prelu_addr;
    int out_addr;

    unsigned int weight_len;
    unsigned int bias_len;
    unsigned int prelu_len;
    void* ptr_conv_weight;
    void* ptr_conv_bias;
    void* ptr_prelu_weight;
};
```

【变量】

变量	含义	类型	取值范围
dev_type	算子运行的设备类型。	unsigned int	暂时不处理
op_type	算子类型。	unsigned int	暂时不处理
blob_id	算子 ID 号。	unsigned int	/
cvt_scale	scale 因子。	unsigned int	/
bits	偏移 bit 数。	unsigned int	/

conv_input_shape_n	输入 batch 数量。	int	/
conv_input_shape_h	输入高度。	int	/
conv_input_shape_w	输入宽度。	int	/
conv_input_shape_c	输出通道数。	int	/
conv_kernel_shape_n	卷积核高度。	int	/
conv_kernel_shape_h	卷积核宽度。	int	/
conv_kernel_shape_w	卷积核通道数。	int	/
conv_kernel_shape_c	卷积核个数。	int	/
conv_stride_x	卷积滑动步长，水平方向。	int	/
conv_stride_y	卷积滑动步长，垂直方向。	int	/
conv_pad	卷积 pad 设置。	int	Left:value &0x000F Right:value &0x00F0 Top:value &0x0F00 Bottom:value &0xF000
conv_dilation	卷积 dilation 设置。	int	/
conv_bn_enbale	bn 使能。	int	0->disable 1->enable BN 和 prelu 不能同时用
conv_bias_mode	偏置模式。	int	0->disable, 1->layer, 2->pre channel, 3->pre elem
conv_act_mode	激活模式。	int	0->disable 1->relu 2->prelu
conv_eltwise_enbale	元素累加使能。	int	/
conv_output_shape_n	卷积输出 batch。	int	/
conv_output_shape_h	卷积输出高度。	int	/
conv_output_shape_w	卷积输出宽度。	int	/
conv_output_shape_c	卷积输出通道数。	int	/
pool_input_shape_n	pool 输入 batch。	int	/
pool_input_shape_h	pool 输入高度。	int	/
pool_input_shape_w	pool 输入宽度。	int	/
pool_input_shape_c	pool 输入通道数。	int	/

pool_kernel	pool 核宽高。	int	宽高相等
pool_pad	pool 核 pad 设置。	int	Left:value &0x000F Right:value &0x00F0 Top:value &0x0F00 Bottom:value &0xF000
pool_stride	pool 核滑动步长设置。	int	X 和 Y 相等
pool_output_shape_n	pool batch 数。	int	/
pool_output_shape_h	pool 输出高度。	int	/
pool_output_shape_w	pool 输出宽度。	int	/
pool_output_shape_c	pool 输出通道数。	int	/
in_addr	输入数据地址。	int	/
weight_addr	卷积权重地址。	int	/
bias_addr	偏置数据地址。	int	/
prelu_addr	激活权重数据地址。	int	/
out_addr	输出数据地址。	int	/
weight_len	权重数据长度，byte。	unsigned int	/
bias_len	偏置数据长度，byte。	unsigned int	/
prelu_len	prelu 数据长度，byte。	unsigned int	/
ptr_conv_weight	权重数据虚拟地址指针。	void*	/
ptr_conv_bias	偏置数据虚拟地址指针。	void*	/
ptr_prelu_weight	prelu 数据虚拟地址指针。	void*	/

【参考头文件】

ipu_runtime.h

【备注】

4.5.ipu_ops

【说明】

算子结构体列表。

【定义】

```
struct ipu_ops
{
    int ipu_ops_num;
```

```
    struct ipu_ops_params* ptr_ops[100];
};
```

【变量】

变量	含义	类型	取值范围
ipu_ops_num	算子数量	int	/
ptr_ops[100]	算子参数指针	struct ipu_ops_params*	/

【参考头文件】

ipu_runtime.h

【备注】

附录

数据格式(ipu_runtime.h)

```
#define FORMAT_T_R8          0：输入图像层，或单通道图像
#define FORMAT_FEATURE      36：常规特征层数据
```

运算完成信号(hw_adaptor.h)

```
#define CONV_ACT_SIG          1：卷积、激活运算完成。
#define CONV_ACT_POOL_SIG     2：卷积、激活、池化运算完成。
```