Sophon Inference Documentation

发布 3.0.0

SOPHGO

2024 年 07 月 04 日

目录

1 声明 **2**

2 快速入门 **4**

2.1 算丰硬件产品概览 . . . 4
2.1.1 算丰加速卡 (SC) 系列 . . . 4
2.1.2 算丰边缘端小盒子 (SE) 系列 . . . 4
2.1.3 算丰服务器 (SA) 系列 . . . 5
2.1.4 算丰模组 (SM) 系列 . . . 5
2.2 算丰软件产品概览 . . . 5
2.2.1 模型部署 . . . 6
2.2.2 SophonSDK . . . 7
2.2.3 Sophon Inference . . . 8
2.2.4 软件安装指南 . . . 8
2.2.4.1 获取软件包并安装链接库 . . . 9
2.2.4.2 离线模型编译工具安装 . . . 9
2.2.4.3 运行时工具安装 . . . 10

3 API 参考 **11**

3.1 SAIL . . . 11
3.2 SAIL C++ API . . . 12
3.2.1 Basic function . . . 12
3.2.2 Data type . . . 12
3.2.3 PaddingAtrr . . . 13
3.2.4 Handle . . . 13
3.2.5 Tensor . . . 14
3.2.6 IOMode . . . 17
3.2.7 Engine . . . 18
3.2.8 BMImage . . . 27
3.2.9 Decoder . . . 28
3.2.10 Bmcv . . . 29
3.3 SAIL Python API . . . 43
3.3.1 Basic function . . . 43
3.3.2 Data type . . . 44
3.3.3 PaddingAtrr . . . 44
3.3.4 sail.Handle . . . 46
3.3.5 sail.IOMode . . . 47
3.3.6 sail.Tensor . . . 47

3.3.7 sail.Engine 51
3.3.8 sail.MultiEngine 58
3.3.9 sail.bm_image 61
3.3.10 sail.BMImage 62
3.3.11 sail.BMImageArray 64
3.3.12 sail.Decoder 65
3.3.13 sail.Bmcv 67

发布记录

CHAPTER 1

声明

法律声明

注意

您购买的产品、服务或特性等应受算能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，算能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。

技术支持

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

邮编 100094

网址 https://www.sophgo.com/

邮箱 sales@sophgo.com

电话 +86-10-57590723 +86-10-57590724

SDK 发布记录

版本	发布日期	说明
V2.0.0	2019.09.20	第一次发布。
V2.0.1	2019.11.16	V2.0.1 版本发布。
V2.0.3	2020.05.07	V2.0.3 版本发布。
V2.2.0	2020.10.12	V2.2.0 版本发布。
V2.3.0	2021.01.11	V2.3.0 版本发布。
V2.3.1	2021.03.09	V2.3.1 版本发布。
V2.3.2	2021.04.01	V2.3.2 版本发布。
V2.4.0	2021.05.23	V2.4.0 版本发布。
V2.5.0	2021.09.02	V2.5.0 版本发布。
V2.6.0	2021.01.30	V2.6.0 版本修正后发布。
V2.7.0	2022.03.16	V2.7.0 版本发布, 20220531 发布补丁版本。
V3.0.0	2022.07.16	V3.0.0 版本发布。

CHAPTER 2

快速入门

2.1 算丰硬件产品概览

2.1.1 算丰加速卡 (SC) 系列

2.1.2 算丰边缘端小盒子 (SE) 系列

2.1.3 算丰服务器 (SA) 系列

2.1.4 算丰模组 (SM) 系列

更多产品请前往官方网站了解：https://www.sophgo.com

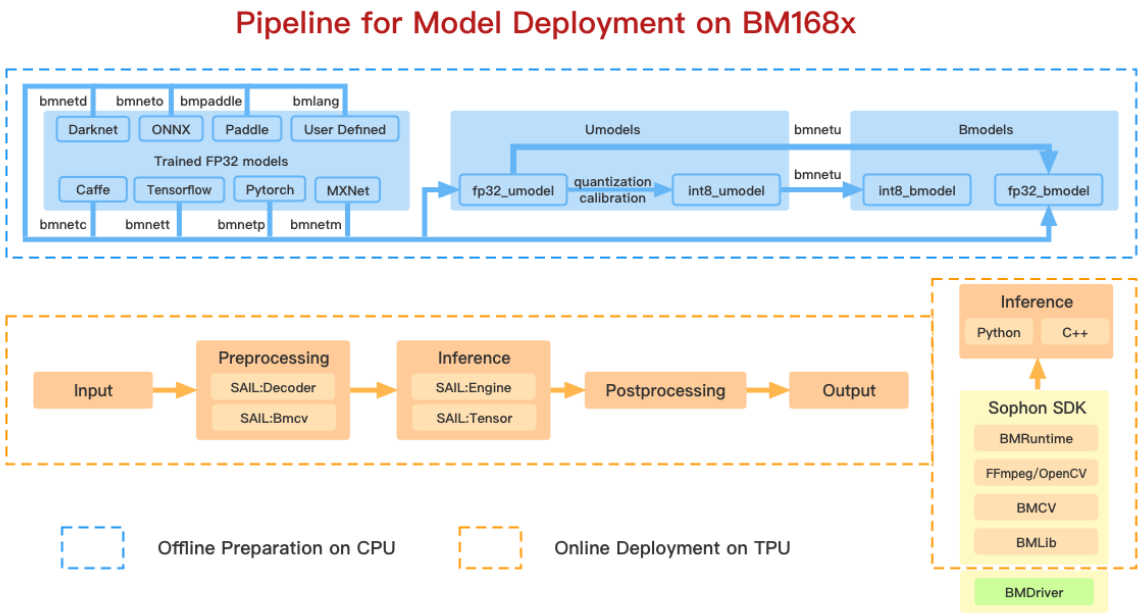
2.2 算丰软件产品概览

针对上一小节中提到的一系列算丰系列 TPU 产品，算能自主研发了一套与之匹配的软件工具：Sophon Software Development Kit(SophonSDK)。

使用算丰 TPU 产品所需的软件栈完全包含在 SophonSDK 中。Sophon Inference 是 SophonSDK 中的一个上层模块，提供了一系列的高级 API 以帮助用户快速部署模型。

在本小节中，我们首先总体讲述基于算丰 TPU 产品的深度学习模型部署的流程。然后，我们分别介绍 SophonSDK 以及 Sophon Inference 的基本概念。最后，我们介绍 SophonSDK 和 Sophon Inference 的安装使用以及相关注意事项。

2.2.1 模型部署



模型部署包含两步：模型的离线编译和在线推理。上图中包含的软件工具都包含在 SophonDK 中。

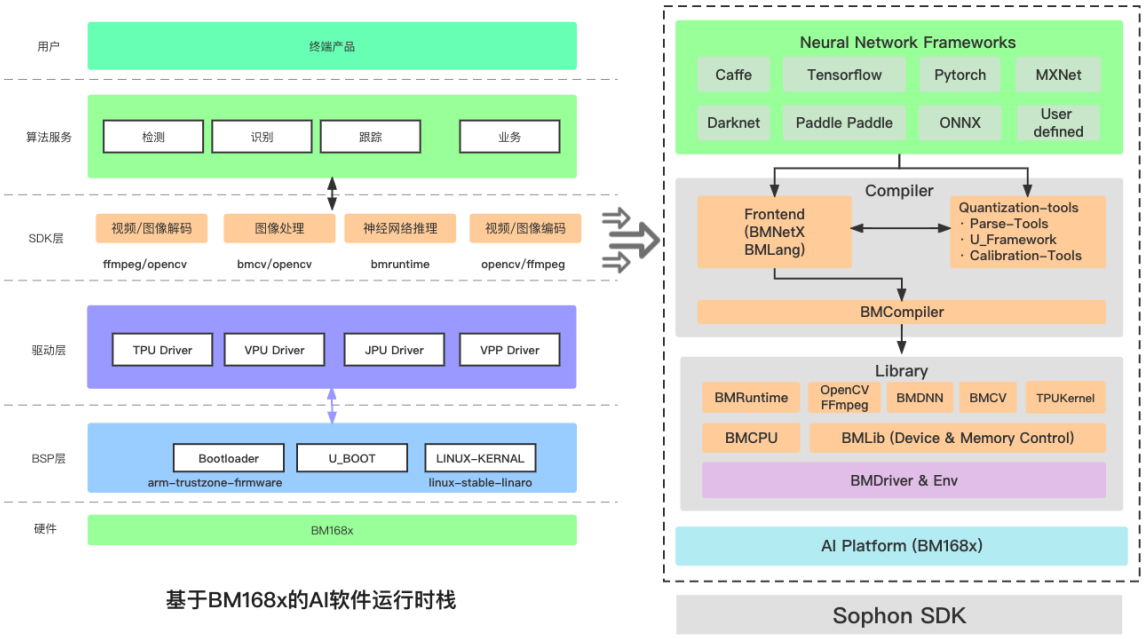
a). 模型离线编译

模型的离线编译对应上图中蓝色虚线框中的内容。这一过程的目的是把用户在各种框架下训练好的深度学习模型转换成我们定义的模型格式：bmodel。目前，算丰 TPU 包含了 FP32 和 INT8 两种类型的计算单元，因此 bmodel 也分为了 fp32_bmodel 和 int8_bmodel。如上图中，经过 tensorflow 训练生成的 xxx.pb 模型可以通过 bmnett 中提供的接口生成 fp32_bmodel；也可以先通过我们提供的脚本先生成 fp32_umodel，再通过量化校准工具生成 int8_umodel，最后通过 bmnetu 生成 int8_bmodel。目前，我们支持 Caffe、Darknet、TensorFlow、PyTorch、MxNet、ONNX、PaddlePaddle 等深度学习框架下的模型编译成 bmodel。这一步骤的完成无需 TPU 的参与，因此是离线的。

b). 模型在线部署

模型的在线部署对应了上图中橙色虚线框中的内容。bmodel 实际上是一系列算丰 TPU 指令的集合。通过使用我们提供的一系列运行时的接口，我们可以把 bmodel 中的指令加载到 TPU 上并执行。将 bmodel 加载到 TPU 内存上这一过程类似于将深度学习模型例如 xxx.pb 加载到主机内存的过程。加载 bmodel 完成之后，使用运行时的接口将输入张量发送给 TPU 并取出计算后得到的输出张量即是模型的推理过程。

2.2.2 SophonSDK



基于BM168x的AI软件运行时栈

SophonSDK 是算能自研的软件包。上图中提及的所有软件模块都包含在 SophonSDK 中，包括了 BMDriver, BMLib, BMCompiler, BMRuntime, Quantization & Calibration Tool, Multimedia(BM_OpenCV & BM_FFmpeg), BMCV, Sophon Inference(SAIL).

具体信息请查看 SophonSDK 入门文档：https://sophgo-doc.gitbook.io/sophonsdk3/

BMDriver：是算丰 TPU 的驱动程序，将会通过 insmod 的方式安装到系统内核中，以源码形式提供。

源码位置：${SDK}/driver

BMLib：提供了一些基础接口，用来控制 TPU 与主机的内存交互。

BMCompiler & BMRuntime：目前多种模型编译工具，支持常见深度学习框架下模型的转换。其中：bmnetc 可以将 Caffe 下训练生成的模型编译成 fp32_bmodel；bmnett 可以将 TensorFlow 下训练生成的模型编译成 fp32_bmodel；bmnetm 可以将 MXNet 下训练生成的模型编译成 fp32_bmodel；bmnetp 可以将 PyTorch 下训练生成的模型编译成 fp32_bmodel；bmnetd 可以将 Darknet 下训练生成的模型编译成 fp32_bmodel；bmneto 可以将 ONNX 下训练生成的模型编译成 fp32_bmodel；bmnpaddle 可以将 PaddlePaddle 下训练生成的模型编译成 fp32_bmodel；bmnetu 可以将 Quantization & Calibration Tool 下生成的 int8_umodel 编译成 int8_bmodel；bmruntime 提供了一些应用接口，用来驱动 TPU 加载 bmodel 并进行模型推理。

Quantization & Calibration Tool：该模块可以将 FP32 精度的模型转换成 INT8 精度的模型

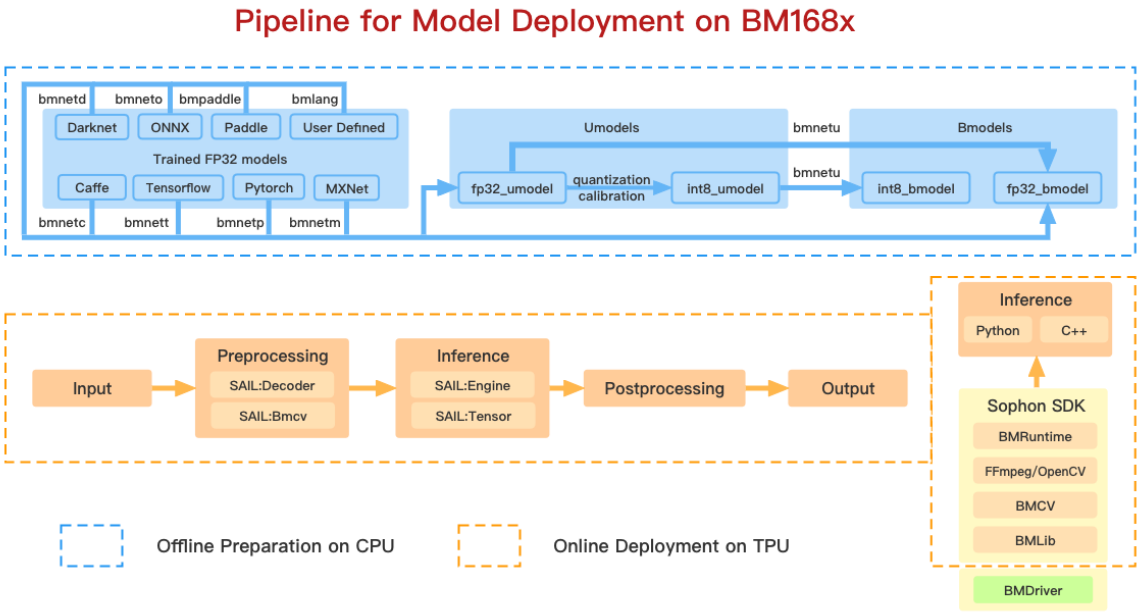
BMMultimedia：提供了一些应用接口，用来驱动 TPU 上的硬件单元进行图像和视频的编解码。

BMCV：提供了一些应用接口，用来驱动 TPU 上的硬件单元进行张量计算和图像处理。

SAIL：提供了一些高级接口，主要是对 BMRuntime、BMCV、BMdecoder 等运行时模块的封装。

BMLang :BMLang 是一种面向 Sophon TPU 的上层编程语言，适用于编写高性能的深度学习、图像处理、矩阵运算等算法程序。

2.2.3 Sophon Inference



Sophon Inference 目前主要包含了 SAIL(Sophon Artificial Intelligent Library) 模块。我们提供了 python 和 c++ 的接口和示例程序，旨在帮助用户快速将模型部署到算丰 TPU 产品上。

SAIL：封装了 BMRuntime, BMCV, BMDecoder 等运行时模块。提供了 python 和 c++ 接口。可用来：

a). 驱动 TPU 加载 bmodel 并进行推理;

b). 驱动 TPU 进行图像和视频的处理。

2.2.4 软件安装指南

在“1.1 算丰硬件产品概览”中，我们介绍了我们目前的四种产品形态：SC、SE、SA、SM。其中，SM 属于定制化的产品，因此在这里不做详细介绍。SC 系列产品为 PCIE 模式的加速卡，作为协处理器接受 X86 主机 CPU 的调用。SE 和 SA 系列产品为 SOC 模式，该模式下，操作系统运行在 TPU 内存上，由 TPU 上的 ARM 处理器负责管理和调度。

对于 SE 和 SA 系列产品的模型部署，我们通常在 X86 系统上编译模型生成 bmodel，再在 SE 和 SA 产品上部署。而在 SE 和 SA 系列产品上，我们已经预装了 SophonSDK 中的全部运行时模块。因此，在这里我们只介绍 X86 主机下 PCIE 模式的 SophonSDK 的安装。如果

你希望你的模型最终运行在 SE 或 SA 产品上，那么只需了解 X86 主机下 SophonSDK 中的离线模型编译工具的安装过程。

2.2.4.1 获取软件包并安装链接库

SophonSDK 以 tar 包的形式发布。命名方式为 sophonsdk3_vx.x.x.tar.gz。其中，3 代表版本号为 3，x.x.x 为详细版本号。解压该软件包后，我们用 ${SDK} 来代替软件包的主目录。

由于 SophonSDK 中存在由不同版本内核编译生成的链接库，因此在解压完之后，我们需要根据当前主机的内核版本来选择适当的链接库。对此，我们提供了对应脚本。每次解压完之后只需运行一次下列命令即可。

```
cd ${SDK}/scripts/
./install_lib.sh nntc
```

2.2.4.2 离线模型编译工具安装

在” 1.2.2 SophonSDK “中我们介绍了 SophonSDK 中的所有软件模块。离线模型编译工具包括了 Quantization & Calibration tool 和 BMCompiler。我们提供了一个脚本来完成离线工具的安装，每次进入终端之后运行以下命令即可完成安装。

```
cd ${SDK}/scripts/
source envsetup_pcie.sh
```

需要注意的是，由于 BMCompiler 依赖有比较多的依赖包，比如 bmnett 依赖 tensorflow，bmnetp 依赖 pytorch，bmnetm 依赖 mxnet。因此，如果你只需要其中某一个工具，可以带参数运行该脚本，如下：

```
cd ${SDK}/scripts/
# 安装 Quantization & Calibration Tool
source envsetup_pcie.sh ufw
# 安装 bmnetu
source envsetup_pcie.sh bmnetu
# 安装 bmnetd
source envsetup_pcie.sh bmnetd
# 安装 bmnett
source envsetup_pcie.sh bmnett
# 安装 bmnetp
source envsetup_pcie.sh bmnetp
# 安装 bmnetm
source envsetup_pcie.sh bmnetm
# 安装 bmnetc
source envsetup_pcie.sh bmnetc
# 安装 bmpaddle
source envsetup_pcie.sh bmpaddle
```

2.2.4.3 运行时工具安装

目前，在 pcie 模式下需要安装的运行时工具只有 BMDriver 和 Sophon Inference 的 python 包，soc 模式已预装，请忽略此步骤。

安装 BMDriver 应当在宿主机上进行，且需要 root 权限，BMDriver 会在主机上编译并安装到系统内核中。

```
cd ${SDK}/scripts/
sudo ./install_driver_pcie.sh
```

安装 Sophon Inference 可以在宿主机上或者提供的基础开发 docker 环境中（推荐方式）进行，需要根据实际环境中的 python 版本选择相应的包安装：

```
# 以python3.8为例
cd ${SDK}/lib/sail/python3/pcie/py38
pip3 install --user sophon-x.x.x-py3-none-any.whl
```

CHAPTER 3

API 参考

3.1 SAIL

SAIL (Sophon Artificial Intelligent Library) 是 Sophon Inference 中的核心模块。SAIL 对 SophonSDK 中的 BMLib、BMDecoder、BMCV、BMRuntime 进行了封装，将 SophonSDK 中原有的 “加载 bmodel 并驱动 TPU 推理”、“驱动 TPU 做图像处理”、“驱动 VPU 做图像和视频解码” 等功能抽象成更为简单的 C++ 接口对外提供；并且使用 pybind11 再次封装，提供简洁易用的 python 接口。

目前，SAIL 模块中所有的类、枚举、函数都在 “sail” 命名空间下，本单元中的文档将向您深入介绍可能用到的 SAIL 中的模块和类。核心的类包括：

- Handle：

SDK 中 BMLib 的 bm_handle_t 的包装类，设备句柄，上下文信息，用来和内核驱动交互信息。

- Tensor：

SDK 中 BMLib 的包装类，封装了对 device memory 的管理以及与 system memory 的同步。

- Engine：

SDK 中 BMRuntime 的包装类，可以加载 bmodel 并驱动 TPU 进行推理。一个 Engine 实例可以加载一个任意的 bmodel，自动地管理输入张量与输出张量对应的内存。

- Decoder

使用 VPU 解码视频，JPU 解码图像，均为硬件解码。

- Bmcv：

SDK 中 BMCV 的包装类，封装了一系列的图像处理函数，可以驱动 TPU 进行图像处理。

3.2 SAIL C++ API

3.2.1 Basic function

1). get_available_tpu_num

```
/** @brief Get the number of available TPUs.
 *
 *  @return Number of available TPUs.
 */
int get_available_tpu_num();
```

****2). set_print_flag ****

```
/** @brief Print main process time use.
 *
 *  @param print_flag.
 */
int set_print_flag(bool print_flag);
```

****3). set_dump_io_flag ****

```
/** @brief Dump input date and output date.
 *
 *  @param dump_io_flag.
 */
int set_dump_io_flag(bool dump_io_flag);
```

****4). get_sail_version ****

```
/** @brief Get Sophon Inference version.
 *
 *  @param sail_version.
 */
void get_sail_version(char* sail_version);
```

****5). set_decoder_env ****

3.2.2 Data type

1). bm_data_type_t

```
enum bm_data_type_t {
  BM_FLOAT32,     // float32
  BM_FLOAT16,     // not supported for now
  BM_INT8,        // int8
  BM_UINT8        // unsigned int8
};
```

3.2.3 PaddingAtrr

1). PaddingAtrr

```
class PaddingAtrr {
public:
    PaddingAtrr(){};
    PaddingAtrr(
        unsigned int crop_start_x,
        unsigned int crop_start_y,
        unsigned int crop_width,
        unsigned int crop_height,
        unsigned char padding_value_r,
        unsigned char padding_value_g,
        unsigned char padding_value_b);
    PaddingAtrr(const PaddingAtrr& other);
    ~PaddingAtrr(){};
    void set_stx(unsigned int stx);
    void set_sty(unsigned int sty);
    void set_w(unsigned int w);
    void set_h(unsigned int h);
    void set_r(unsigned int r);
    void set_g(unsigned int g);
    void set_b(unsigned int b);

    unsigned int    dst_crop_stx; // Offset x information relative to the origin of dst image
    unsigned int    dst_crop_sty; // Offset y information relative to the origin of dst image
    unsigned int    dst_crop_w;   // The width after resize
    unsigned int    dst_crop_h;   // The height after resize
    unsigned char   padding_r;    // Pixel value information of R channel
    unsigned char   padding_g;    // Pixel value information of G channel
    unsigned char   padding_b;    // Pixel value information of B channel
};
```

3.2.4 Handle

1). Handle Constructor

```
/**
 * @brief Constructor using existed bm_handle_t.
 *
 * @param handle A bm_handle_t
 */
Handle(bm_handle_t handle);

/**
 * @brief Constructor with device id.
 *
 * @param dev_id Device id
 */
Handle(int dev_id);
```

2). data

```
/**
 *  @brief Get inner bm_handle_t.
 *
 *  @return Inner bm_handle_t
 */
bm_handle_t data();
```

3). get_device_id

```
/**
* @brief Get device id of this handle.
*
* @return Device id.
*/
int get_device_id();
```

4). get_sn

```
/**
* @brief Get serial number
*
* @return serial number
*/
std::string get_sn();
```

3.2.5 Tensor

1). Tensor Constructor

```
/**
 * @brief Common constructor.
 * @detail
 *  case 0: only allocate system memory
 *          (handle, shape, dtype, true, false)
 *  case 1: only allocate device memory
 *          (handle, shape, dtype, false, true)
 *  case 2: allocate system memory and device memory
 *          (handle, shape, dtype, true, true)
 *
 * @param handle       Handle instance
 * @param shape        Shape of the tensor
 * @param own_sys_data Indicator of whether own system memory.
 * @param own_dev_data Indicator of whether own device memory.
 */
explicit Tensor(
    Handle                handle,
    const std::vector<int>& shape,
    bm_data_type_t        dtype,
```

(下页继续)

(续上页)

```
    bool                own_sys_data,
    bool                own_dev_data);

/**
 *  @brief Copy constructor.
 *
 *  @param tensor A Tensor instance
 */
Tensor(const Tensor& tensor);
```

2). Tensor Assign Function

```
/**
 * @brief Assignment function.
 *
 * @param tensor A Tensor instance
 * @return A Tensor instance
 */
Tensor& operator=(const Tensor& tensor);
```

3). shape

```
/**
 * @brief Get shape of the tensor.
 *
 * @return Shape of the tensor
 */
const std::vector<int>& shape() const;
```

4). dtype

```
/**
 * @brief Get data type of the tensor.
 *
 * @return Data type of the tensor
 */
void dtype();
```

5). reshape

```
/**
 * @brief Reset shape of the tensor.
 *
 * @param shape Shape of the tensor
 */
void reshape(const std::vector<int>& shape);
```

6). own_sys_data

```
/**
 * @brief Judge if the tensor owns data in system memory.
 *
 * @return True for owns data in system memory.
 */
bool own_sys_data();
```

7). own_dev_data

```
/**
 * @brief Judge if the tensor owns data in device memory.
 *
 * @return True for owns data in device memory.
 */
bool own_dev_data();
```

8). sys_data

```
/**
 * @brief Get data pointer in system memory of the tensor.
 *
 * @return Data pointer in system memory of the tensor
 */
void* sys_data();
```

9). dev_data

```
/**
 * @brief Get pointer to device memory of the tensor.
 *
 * @return Pointer to device memory of the tensor
 */
bm_device_mem_t* dev_data();
```

10). reset_sys_data

```
/**
 * @brief Reset data pointer in system memory of the tensor.
 *
 * @param data  Data pointer in system memory of the tensor
 * @param shape Shape of the data
 */
void reset_sys_data(
    void*              data,
    std::vector<int>& shape);
```

11). reset_dev_data

```
/**
 * @brief Reset pointer to device memory of the tensor.
 *
```

(下页继续)

(续上页)

```
 * @param data Pointer to device memory
 */
void reset_dev_data(bm_device_mem_t* data);
```

12). sync_s2d

```
/**
 * @brief Copy data from system memory to device memory.
 */
void sync_s2d();

/**
 * @brief Copy data from system memory to device memory with specified size.
 *
 * @param size Byte size to be copied
 */
void sync_s2d(int size);
```

13). sync_d2s

```
/**
 * @brief Copy data from device memory to system memory.
 */
void sync_d2s();

/**
 * @brief Copy data from device memory to system memory with specified size.
 *
 * @param size Byte size to be copied
 */
void sync_d2s(int size);
```

14). free

```
/**
 * @brief Free system and device memroy of the tensor.
 */
void free();
```

3.2.6 IOMode

1). IOMode

```
enum IOMode {
  /// Input tensors are in system memory while output tensors are
  /// in device memory.
  SYSI,
  /// Input tensors are in device memory while output tensors are
  /// in system memory.
```

(下页继续)

(续上页)

```
  SYSO,
  /// Both input and output tensors are in system memory.
  SYSIO,
  /// Both input and output tensors are in device memory.
  DEVIO
};
```

3.2.7 Engine

1). Engine Constructor

```
/**
 * @brief Constructor does not load bmodel.
 *
 * @param tpu_id TPU ID. You can use bm-smi to see available IDs.
 */
 Engine(int tpu_id);

/**
 * @brief Constructor does not load bmodel.
 *
 * @param handle Handle created elsewhere.
 */
Engine(const Handle&   handle);

/**
 * @brief Constructor loads bmodel from file.
 *
 * @param bmodel_path Path to bmodel
 * @param tpu_id      TPU ID. You can use bm-smi to see available IDs.
 * @param mode        Specify the input/output tensors are in system memory
 *                    or device memory
 */
Engine(
    const std::string& bmodel_path,
    int                tpu_id,
    IOMode             mode);

/**
 * @brief Constructor loads bmodel from file.
 *
 * @param bmodel_path Path to bmodel
 * @param handle      Handle created elsewhere.
 * @param mode        Specify the input/output tensors are in system memory
 *                    or device memory
 */
Engine(
    const std::string& bmodel_path,
    const Handle&      handle,
```

(下页继续)

(续上页)

```
    IOMode              mode);

/**
 * @brief Constructor loads bmodel from system memory.
 *
 * @param bmodel_ptr  Pointer to bmodel in system memory
 * @param bmodel_size Byte size of bmodel in system memory
 * @param tpu_id      TPU ID. You can use bm-smi to see available IDs.
 * @param mode        Specify the input/output tensors are in system memory
 *                    or device memory
 */
 Engine(
    const void* bmodel_ptr,
    size_t      bmodel_size,
    int         tpu_id,
    IOMode      mode);

/**
 * @brief Constructor loads bmodel from system memory.
 *
 * @param bmodel_ptr  Pointer to bmodel in system memory
 * @param bmodel_size Byte size of bmodel in system memory
 * @param handle      Handle created elsewhere.
 * @param mode        Specify the input/output tensors are in system memory
 *                    or device memory
 */
Engine(
    const void*        bmodel_ptr,
    size_t             bmodel_size,
    const Handle&      handle,
    IOMode             mode);

/**
 * @brief Copy constructor.
 *
 * @param other An other Engine instance.
 */
Engine(const Engine& other);
```

2). Engine Assign Function

```
/**
 * @brief Assignment function.
 *
 * @param other An other Engine instance.
 * @return Reference of a Engine instance.
 */
Engine<Dtype>& operator=(const Engine& other);
```

3). get_handle

```
/**
 * @brief Get Handle instance.
 *
 * @return Handle instance
 */
Handle get_handle();
```

4). load

```
/**
 * @brief Load bmodel from file.
 *
 * @param bmodel_path Path to bmodel
 * @return Program state
 *     @retval true  Success
 *     @retval false Failure
 */
bool load(const std::string& bmodel_path);

/**
 * @brief Load bmodel from system memory.
 *
 * @param bmodel_ptr  Pointer to bmodel in system memory
 * @param bmodel_size Byte size of bmodel in system memory
 * @return Program state
 *     @retval true  Success
 *    @retval false Failure
 */
bool load(const void* bmodel_ptr, size_t bmodel_size);
```

5). get_graph_names

```
/**
 * @brief Get all graph names in the loaded bomodels.
 *
 * @return All graph names
 */
std::vector<std::string> get_graph_names();
```

6). set_io_mode

```
/**
 * @brief Set IOMode for a graph.
 *
 * @param graph_name The specified graph name
 * @param mode The specified IOMode
 */
void set_io_mode(
  const std::string& graph_name,
  IOMode             mode);
```

7). get_input_names

```
/**
 * @brief Get all input tensor names of the specified graph.
 *
 * @param graph_name The specified graph name
 * @return All the input tensor names of the graph
 */
std::vector<std::string> get_input_names(const std::string& graph_name);
```

8). get_output_names

```
/**
 * @brief Get all output tensor names of the specified graph.
 *
 * @param graph_name The specified graph name
 * @return All the output tensor names of the graph
 */
std::vector<std::string> get_output_names(const std::string& graph_name);
```

9). get_max_input_shapes

```
/**
 * @brief Get max shapes of input tensors in a graph.
 *
 * For static models, the max shape is fixed and it should not be changed.
 * For dynamic models, the tensor shape should be smaller than or equal to
 * the max shape.
 *
 * @param graph_name The specified graph name
 * @return Max shape of input tensors
 */
std::map<std::string, std::vector<int>> get_max_input_shapes(
    const std::string& graph_name);
```

10). get_input_shape

```
/**
 * @brief Get the shape of an input tensor in a graph.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return The shape of the tensor
 */
std::vector<int> get_input_shape(
    const std::string& graph_name,
    const std::string& tensor_name);
```

11). get_max_output_shapes

```
/**
 * @brief Get max shapes of output tensors in a graph.
 *
```

(下页继续)

(续上页)

```
 * For static models, the max shape is fixed and it should not be changed.
 * For dynamic models, the tensor shape should be smaller than or equal to
 * the max shape.
 *
 * @param graph_name The specified graph name
 * @return Max shape of output tensors
 */
std::map<std::string, std::vector<int>> get_max_output_shapes(
    const std::string& graph_name);
```

12). get_output_shape

```
/**
 * @brief Get the shape of an output tensor in a graph.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return The shape of the tensor
 */
std::vector<int> get_output_shape(
    const std::string& graph_name,
    const std::string& tensor_name);
```

13). get_input_dtype

```
/**
 * @brief Get data type of an input tensor. Refer to bmdef.h as following.
 *   typedef enum {
 *     BM_FLOAT32 = 0,
 *     BM_FLOAT16 = 1,
 *     BM_INT8 = 2,
 *     BM_UINT8 = 3,
 *     BM_INT16 = 4,
 *     BM_UINT16 = 5,
 *     BM_INT32 = 6,
 *     BM_UINT32 = 7
 *   } bm_data_type_t;
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return Data type of the input tensor
 */
bm_data_type_t get_input_dtype(
    const std::string& graph_name,
    const std::string& tensor_name);
```

14). get_output_dtype

```
/**
 * @brief Get data type of an output tensor. Refer to bmdef.h as following.
```

(下页继续)

(续上页)

```
*   typedef enum {
*     BM_FLOAT32 = 0,
*     BM_FLOAT16 = 1,
*     BM_INT8 = 2,
*     BM_UINT8 = 3,
*     BM_INT16 = 4,
*     BM_UINT16 = 5,
*     BM_INT32 = 6,
*     BM_UINT32 = 7
*   } bm_data_type_t;
*
* @param graph_name  The specified graph name
* @param tensor_name The specified tensor name
* @return Data type of the input tensor
*/
bm_data_type_t get_output_dtype(
    const std::string& graph_name,
    const std::string& tensor_name);
```

15). get_input_scale

```
/**
 * @brief Get scale of an input tensor. Only used for int8 models.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return Scale of the input tensor
 */
float get_input_scale(
    const std::string& graph_name,
    const std::string& tensor_name);
```

16). get_output_scale

```
/**
 * @brief Get scale of an output tensor. Only used for int8 models.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return Scale of the output tensor
 */
float get_output_scale(
    const std::string& graph_name,
    const std::string& tensor_name);
```

17). reshape

```
/**
 * @brief Reshape input tensor for dynamic models.
 *
```

(下页继续)

(续上页)

```
 * The input tensor shapes may change when running dynamic models.
 * New input shapes should be set before inference.
 *
 * @param graph_name   The specified graph name
 * @param input_shapes Specified shapes of all input tensors of the graph
 * @return 0 for success and 1 for failure
 */
int reshape(
    const std::string&                        graph_name,
    std::map<std::string, std::vector<int>>& input_shapes);
```

18). get_input_tensor

```
/**
 * @brief Get the specified input tensor.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return The specified input tensor
 */
Tensor* get_input_tensor(
    const std::string& graph_name,
    const std::string& tensor_name);
```

19). get_output_tensor

```
/**
 * @brief Get the specified output tensor.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @return The specified output tensor
 */
Tensor* get_output_tensor(
    const std::string& graph_name,
    const std::string& tensor_name);
```

20). scale_input_tensor

```
/**
 * @brief Scale input tensor for int8 models.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @param data        Pointer to float data to be scaled
 */
void scale_input_tensor(
    const std::string& graph_name,
    const std::string& tensor_name,
    float*             data);
```

21). scale_output_tensor

```
/**
 * @brief Scale output tensor for int8 models.
 *
 * @param graph_name  The specified graph name
 * @param tensor_name The specified tensor name
 * @param data        Pointer to float data to be scaled
 */
void scale_output_tensor(
    const std::string& graph_name,
    const std::string& tensor_name,
    float*             data);
```

22). scale_fp32_to_int8

```
/**
 * @brief Scale data from float32 to int8. Only used for int8 models.
 *
 * @param src   Poniter to float32 data
 * @param dst   Poniter to int8 data
 * @param scale Value of scale
 * @param size  Size of data
 */
void scale_fp32_to_int8(float* src, int8_t* dst, float scale, int size);
```

23). scale_int8_to_fp32

```
/**
 * @brief Scale data from int8 to float32. Only used for int8 models.
 *
 * @param src   Poniter to int8 data
 * @param dst   Poniter to float32 data
 * @param scale Value of scale
 * @param size  Size of data
 */
void scale_int8_to_fp32(int8_t* src, float* dst, float scale, int size);
```

24). process

```
/**
 * @brief Inference with builtin input and output tensors.
 *
 * @param graph_name The specified graph name
 */
void process(const std::string& graph_name);

/**
 * @brief Inference with provided input tensors.
 *
 * @param graph_name    The specified graph name
 * @param input_shapes  Shapes of all input tensors
```

(下页继续)

(续上页)

```
 * @param input_tensors Data pointers of all input tensors in system memory
 */
void process(
    const std::string&                      graph_name,
    std::map<std::string, std::vector<int>>& input_shapes,
    std::map<std::string, void*>&           input_tensors);

/**
 * @brief Inference with provided input and output tensors.
 *
 * @param graph_name The specified graph name
 * @param input      Input tensors
 * @param output     Output tensors
 */
void process(
    const std::string&             graph_name,
    std::map<std::string, Tensor*>& input,
    std::map<std::string, Tensor*>& output);

/**
 * @brief Inference with provided input and output tensors and input shapes.
 *
 * @param graph_name   The specified graph name
 * @param input        Input tensors
 * @param input_shapes Real input tensor shapes
 * @param output       Output tensors
 */
void process(
    const std::string&                      graph_name,
    std::map<std::string, Tensor*>&         input,
    std::map<std::string, std::vector<int>>& input_shapes,
    std::map<std::string, Tensor*>&         output);
```

25). create_input_tensors_map

```
/**
 * @brief Create input tensors map, according to and bmodel.
 * @param graph_name   The specified graph name
 * @param create_mode Tensor Create mode
 *  case 0: only allocate system memory
 *  case 1: only allocate device memory
 *  case other: according to engine IOMode
 */
std::map<std::string, Tensor*> create_input_tensors_map(
    const std::string& graph_name,
    int create_mode = -1);
```

26). create_output_tensors_map

```
/**
```

(下页继续)

(续上页)

```
 * @brief Create output tensors map, according to and bmodel.
 * @param graph_name   The specified graph name
 * @param create_mode Tensor Create mode
 *  case 0: only allocate system memory
 *  case 1: only allocate device memory
 *  case other: according to engine IOMode
 */
std::map<std::string, Tensor*> create_output_tensors_map(
    const std::string& graph_name,
    int create_mode = -1);
```

3.2.8 BMImage

1). BMImage Constructor

```
/**
 * @brief The default Constructor.
 */
BMImage();

/**
 * @brief The BMImage Constructor.
 *
 * @param handle A Handle instance
 * @param h      Image width
 * @param w      Image height
 * @param format Image format
 * @param dtype  Data type
 */
BMImage(
    Handle&                 handle,
    int                     h,
    int                     w,
    bm_image_format_ext     format,
    bm_image_data_format_ext dtype);
```

2). data

```
/**
 * @brief Get inner bm_image
 *
 * @return The inner bm_image
 */
bm_image& data();
```

3). width

```
/**
 * @brief Get the img width.
```

(下页继续)

(续上页)

```
 *
 * @return the width of img
 */
int width();
```

4). height

```
/**
 * @brief Get the img height.
 *
 * @return the height of img
 */
int height();
```

5). format

```
/**
 * @brief Get the img format.
 *
 * @return the format of img
 */
bm_image_format_ext format();
```

3.2.9 Decoder

1). Decoder Constructor

```
/**
 * @brief Constructor.
 *
 * @param file_path  Path or rtsp url to the video/image file.
 * @param compressed Whether the format of decoded output is compressed NV12.
 * @param tpu_id     ID of TPU, there may be more than one TPU for PCIE mode.
 */
Decoder(
    const std::string& file_path,
    bool               compressed = true,
    int                tpu_id = 0);
```

2). is_opened

```
/**
 * @brief Judge if the source is opened successfully.
 *
 * @return True if the source is opened successfully
 */
bool is_opened();
```

3). read

```
/**
 * @brief Read a bm_image from the Decoder.
 *
 * @param handle A bm_handle_t instance
 * @param image Reference of bm_image to be read to
 * @return 0 for success and 1 for failure
 */
int read(Handle& handle, bm_image& image);

/**
 * @brief Read a BMImage from the Decoder.
 *
 * @param handle A bm_handle_t instance
 * @param image Reference of BMImage to be read to
 * @return 0 for success and 1 for failure
 */
int read(Handle& handle, BMImage& image);
```

3.2.10 Bmcv

1). Bmcv Constructor

```
/**
 * @brief Constructor.
 *
 * @param handle A Handle instance
 */
explicit Bmcv(Handle handle);
```

2). bm_image_to_tensor

```
/**
 * @brief Convert BMImage to tensor.
 *
 * @param img    Input image
 * @param tensor Output tensor
 */
void bm_image_to_tensor(BMImage &img, Tensor &tensor);

/**
 * @brief Convert BMImage to tensor.
 *
 * @param img Input image
 */
Tensor bm_image_to_tensor(BMImage &img);
```

3). tensor_to_bm_image

```
/**
 * @brief Convert tensor to BMImage.
```

(下页继续)

(续上页)

```
 *
 * @param tensor   Input tensor
 * @param img      Output image
 */
void tensor_to_bm_image(Tensor &tensor, BMImage &img);

/**
 * @brief Convert tensor to BMImage.
 *
 * @param tensor   Input tensor
 */
BMImage tensor_to_bm_image(Tensor &tensor);
```

4). crop_and_resize

```
/**
 * @brief Crop then resize an image.
 *
 * @param input    Input image
 * @param output   Output image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @param resize_w Target width
 * @param resize_h Target height
 * @return 0 for success and other for failure
 */
int crop_and_resize(
    BMImage                 &input,
    BMImage                 &output,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    int                     resize_w,
    int                     resize_h);

/**
 * @brief Crop then resize an image.
 *
 * @param input    Input image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @param resize_w Target width
 * @param resize_h Target height
 * @return Output image
 */
```

(下页继续)

(续上页)

```
BMImage crop_and_resize(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    int                     resize_w,
    int                     resize_h);
```

5). crop

```
/**
 * @brief Crop an image with given window.
 *
 * @param input    Input image
 * @param output   Output image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @return 0 for success and other for failure
 */
int crop(
    BMImage                 &input,
    BMImage                 &output,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h);

/**
 * @brief Crop an image with given window.
 *
 * @param input    Input image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @return Output image
 */
 BMImage crop(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h);
```

6). resize

```
/**
```

(下页继续)

(续上页)

```
 * @brief Resize an image with interpolation of INTER_NEAREST.
 *
 * @param input    Input image
 * @param output   Output image
 * @param resize_w Target width
 * @param resize_h Target height
 * @return 0 for success and other for failure
 */
int resize(
    BMImage                    &input,
    BMImage                    &output,
    int                        resize_w,
    int                        resize_h);

/**
 * @brief Resize an image with interpolation of INTER_NEAREST.
 *
 * @param input    Input image
 * @param resize_w Target width
 * @param resize_h Target height
 * @return Output image
 */
BMImage resize(
    BMImage                    &input,
    int                        resize_w,
    int                        resize_h);
```

7). vpp_crop_and_resize

```
/**
 * @brief Crop then resize an image using vpp.
 *
 * @param input    Input image
 * @param output   Output image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @param resize_w Target width
 * @param resize_h Target height
 * @return 0 for success and other for failure
 */
 int vpp_crop_and_resize(
     BMImage                    &input,
     BMImage                    &output,
     int                        crop_x0,
     int                        crop_y0,
     int                        crop_w,
     int                        crop_h,
     int                        resize_w,
```

(下页继续)

(续上页)

```
    int                     resize_h);

/**
 * @brief Crop then resize an image using vpp.
 *
 * @param input    Input image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @param resize_w Target width
 * @param resize_h Target height
 * @return Output image
 */
BMImage vpp_crop_and_resize(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    int                     resize_w,
    int                     resize_h);
```

8). vpp_crop_and_resize_padding

```
/**
 * @brief Crop then resize an image using vpp.
 *
 * @param input       Input image
 * @param output      Output image
 * @param crop_x0     Start point x of the crop window
 * @param crop_y0     Start point y of the crop window
 * @param crop_w      Width of the crop window
 * @param crop_h      Height of the crop window
 * @param resize_w    Target width
 * @param resize_h    Target height
 * @param padding_in  PaddingAtrr info
 * @return 0 for success and other for failure
 */
int vpp_crop_and_resize_padding(
    BMImage                 &input,
    BMImage                 &output,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    int                     resize_w,
    int                     resize_h,
    PaddingAtrr             &padding_in);
```

(下页继续)

(续上页)

```
/**
 * @brief Crop then resize an image using vpp.
 *
 * @param input       Input image
 * @param crop_x0     Start point x of the crop window
 * @param crop_y0     Start point y of the crop window
 * @param crop_w      Width of the crop window
 * @param crop_h      Height of the crop window
 * @param resize_w    Target width
 * @param resize_h    Target height
 * @param padding_in  PaddingAtrr info
 * @return Output image
 */
BMImage vpp_crop_and_resize_padding(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    int                     resize_w,
    int                     resize_h,
    PaddingAtrr             &padding_in);
```

9). vpp_crop

```
/**
 * @brief Crop an image with given window using vpp.
 *
 * @param input    Input image
 * @param output   Output image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @return 0 for success and other for failure
 */
int vpp_crop(
    BMImage                 &input,
    BMImage                 &output,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h);

/**
 * @brief Crop an image with given window using vpp.
 *
 * @param input    Input image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
```

(下页继续)

(续上页)

```
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @return Output image
 */
BMImage vpp_crop(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h);
```

10). vpp_crop_padding

```
/**
 * @brief Crop an image with given window using vpp.
 *
 * @param input       Input image
 * @param output      Output image
 * @param crop_x0     Start point x of the crop window
 * @param crop_y0     Start point y of the crop window
 * @param crop_w      Width of the crop window
 * @param crop_h      Height of the crop window
 * @param padding_in  PaddingAtrr info
 * @return 0 for success and other for failure
 */
int vpp_crop_padding(
    BMImage                 &input,
    BMImage                 &output,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
    int                     crop_h,
    PaddingAtrr             &padding_in);

/**
 * @brief Crop an image with given window using vpp.
 *
 * @param input    Input image
 * @param crop_x0  Start point x of the crop window
 * @param crop_y0  Start point y of the crop window
 * @param crop_w   Width of the crop window
 * @param crop_h   Height of the crop window
 * @param padding_in  PaddingAtrr info
 * @return Output image
 */
BMImage vpp_crop_padding(
    BMImage                 &input,
    int                     crop_x0,
    int                     crop_y0,
    int                     crop_w,
```

(下页继续)

(续上页)

```
    int                 crop_h,
    PaddingAtrr         &padding_in);
```

11). vpp_resize

```
/**
 * @brief Resize an image with interpolation of INTER_NEAREST using vpp.
 *
 * @param input    Input image
 * @param output   Output image
 * @param resize_w Target width
 * @param resize_h Target height
 * @return 0 for success and other for failure
 */
int vpp_resize(
    BMImage             &input,
    BMImage             &output,
    int                 resize_w,
    int                 resize_h);

/**
 * @brief Resize an image with interpolation of INTER_NEAREST using vpp.
 *
 * @param input    Input image
 * @param resize_w Target width
 * @param resize_h Target height
 * @return Output image
 */
BMImage vpp_resize(
    BMImage             &input,
    int                 resize_w,
    int                 resize_h);
```

12). vpp_resize_padding

```
/**
 * @brief Resize an image with interpolation of INTER_NEAREST using vpp.
 *
 * @param input       Input image
 * @param output      Output image
 * @param resize_w    Target width
 * @param resize_h    Target height
 * @param padding_in  PaddingAtrr info
 * @return 0 for success and other for failure
 */
int vpp_resize_padding(
    BMImage             &input,
    BMImage             &output,
    int                 resize_w,
    int                 resize_h,
```

(下页继续)

(续上页)

```
    PaddingAtrr                &padding_in);

/**
 * @brief Resize an image with interpolation of INTER_NEAREST using vpp.
 *
 * @param input       Input image
 * @param resize_w    Target width
 * @param resize_h    Target height
 * @param padding_in  PaddingAtrr info
 * @return Output image
 */
BMImage vpp_resize_padding(
    BMImage                    &input,
    int                        resize_w,
    int                        resize_h,
    PaddingAtrr                &padding_in);
```

13). warp

```
/**
 * @brief Applies an affine transformation to an image.
 *
 * @param input    Input image
 * @param output   Output image
 * @param matrix   2x3 transformation matrix
 * @return 0 for success and other for failure
 */
int warp(
    BMImage                        &input,
    BMImage                        &output,
    const std::pair<
      std::tuple<float, float, float>,
      std::tuple<float, float, float>> &matrix);

/**
 * @brief Applies an affine transformation to an image.
 *
 * @param input    Input image
 * @param matrix   2x3 transformation matrix
 * @return Output image
 */
BMImage warp(
    BMImage                        &input,
    const std::pair<
      std::tuple<float, float, float>,
      std::tuple<float, float, float>> &matrix);
```

14). convert_to

```
/**
```

(下页继续)

(续上页)

```
 * @brief Applies a linear transformation to an image.
 *
 * @param input        Input image
 * @param output       Output image
 * @param alpha_beta   (a0, b0), (a1, b1), (a2, b2) factors
 * @return 0 for success and other for failure
 */
int convert_to(
    BMImage                     &input,
    BMImage                     &output,
    const std::tuple<
      std::pair<float, float>,
      std::pair<float, float>,
      std::pair<float, float>>   &alpha_beta);

/**
 * @brief Applies a linear transformation to an image.
 *
 * @param input        Input image
 * @param alpha_beta   (a0, b0), (a1, b1), (a2, b2) factors
 * @return Output image
 */
BMImage convert_to(
    BMImage                     &input,
    const std::tuple<
      std::pair<float, float>,
      std::pair<float, float>,
      std::pair<float, float>>   &alpha_beta);
```

15). yuv2bgr

```
/**
 * @brief Convert an image from YUV to BGR.
 *
 * @param input    Input image
 * @param output   Output image
 * @return 0 for success and other for failure
 */
int yuv2bgr(
    BMImage                     &input,
    BMImage                     &output);

/**
 * @brief Convert an image from YUV to BGR.
 *
 * @param input    Input image
 * @return Output image
 */
BMImage yuv2bgr(BMImage  &input);
```

16). vpp_convert

```
/**
 * @brief Convert an image to BGR PLANAR format using vpp.
 *
 * @param input    Input image
 * @param output   Output image
 * @return 0 for success and other for failure
 */
int vpp_convert(
    BMImage  &input,
    BMImage  &output);

/**
 * @brief Convert an image to BGR PLANAR format using vpp.
 *
 * @param input    Input image
 * @return Output image
 */
BMImage vpp_convert(BMImage  &input);
```

17). convert

```
/**
 * @brief Convert an image to BGR PLANAR format.
 *
 * @param input    Input image
 * @param output   Output image
 * @return 0 for success and other for failure
 */
int convert(
    BMImage  &input,
    BMImage  &output);

/**
 * @brief Convert an image to BGR PLANAR format.
 *
 * @param input    Input image
 * @return Output image
 */
BMImage convert(BMImage  &input);
```

18). rectangle

```
/**
 * @brief Draw a rectangle on input image.
 *
 * @param image      Input image
 * @param x0         Start point x of rectangle
 * @param y0         Start point y of rectangle
 * @param w          Width of rectangle
 * @param h          Height of rectangle
 * @param color      Color of rectangle
```

(下页继续)

(续上页)

```
 * @param thickness  Thickness of rectangle
 * @return 0 for success and other for failure
 */
int rectangle(
    BMImage                     &image,
    int                         x0,
    int                         y0,
    int                         w,
    int                         h,
    const std::tuple<int, int, int> &color,
    int                         thickness=1);
```

19). rectangle_

20). imwrite

```
/**
 * @brief Save the image to the specified file.
 *
 * @param filename   Name of the file
 * @param image      Image to be saved
 * @return 0 for success and other for failure
 */
int imwrite(
    const std::string &filename,
    BMImage           &image);
```

21). imwrite_

```
/**
 * @brief Save the image to the specified file.
 *
 * @param filename   Name of the file
 * @param image      Image to be saved
 * @return 0 for success and other for failure
 */
int imwrite_(
    const std::string &filename,
    const bm_image    &image);
```

22). get_handle

```
/**
 * @brief Get Handle instance.
 *
 * @return Handle instance
 */
Handle get_handle();
```

23). putText

```
/**
 * @brief put text on input image
 *
 * @param image     Input image
 * @param text      Text string to be drawn
 * @param x         Start x
 * @param y         Start y
 * @param color     Color of text
 * @param fontScale Font scale factor that is multiplied by the font-specific base size
 * @param thickness Thickness of the lines used to draw a text
 * @return int
 */
int putText(
    const BMImage                &image,
    const std::string            &text,
    int                          x,
    int                          y,
    const std::tuple<int, int, int> &color, // BGR
    float                        fontScale,
    int                          thickness=1);
```

24). putText_

```
/**
 * @brief put text on input image
 *
 * @param image     Input image
 * @param text      Text string to be drawn
 * @param x         Start x
 * @param y         Start y
 * @param color     Color of text
 * @param fontScale Font scale factor that is multiplied by the font-specific base size
 * @param thickness Thickness of the lines used to draw a text
 * @return int
 */
int putText_(
    const bm_image               &image,
    const std::string            &text,
    int                          x,
    int                          y,
    const std::tuple<int, int, int> &color, // BGR
    float                        fontScale,
    int                          thickness=1);
```

25). image_add_weighted

```
/**
 * @brief output = input1 * alpha + input2 * beta + gamma
 */
int image_add_weighted(
    BMImage          &input1,
```

(下页继续)

(续上页)

```
    float           alpha,
    BMImage         &input2,
    float           beta,
    float           gamma,
    BMImage         &output);

BMImage image_add_weighted(
    BMImage         &input1,
    float           alpha,
    BMImage         &input2,
    float           beta,
    float           gamma);
```

26). image_add_weighted

```
/**
 * @brief Copy input image to output
 * @param input   Input image
 * @param output  Output image
 * @param start_x Target starting point x
 * @param start_y Target starting point y
 */
int image_copy_to(
    bm_image input,
    bm_image output,
    int start_x,
    int start_y);

int image_copy_to(
    BMImage &input,
    BMImage &output,
    int start_x = 0,
    int start_y = 0);
```

26). image_add_weighted

```
/**
 * @brief Copy input image to output with padding
 * @param input   Input image
 * @param output  Output image
 * @param start_x       Target starting point x
 * @param start_y       Target starting point y
 * @param padding_r     padding value of r
 * @param padding_g     padding value of g
 * @param padding_b     padding value of b
*/
int image_copy_to_padding(
    bm_image input,
    bm_image output,
    unsigned int padding_r,
```

(下页继续)

(续上页)

```
    unsigned int padding_g,
    unsigned int padding_b,
    int start_x,
    int start_y);

int image_copy_to_padding(
    BMImage &input,
    BMImage &output,
    unsigned int padding_r,
    unsigned int padding_g,
    unsigned int padding_b,
    int start_x = 0,
    int start_y = 0);
```

27). image_add_weighted

3.3 SAIL Python API

SAIL use "pybind11" to wrap python interfaces, support python3.5, python3.6, python3.7, python3.8

3.3.1 Basic function

```
def get_available_tpu_num():
    """ Get the number of available TPUs.

    Returns
    -------
    tpu_num : int
        Number of available TPUs
    """

def set_print_flag(print_flag):
    """ Print main process time use.

    Parameters
    ----------
    print_flag : bool
        if print_flag is true, print main process time use, Otherwise not print.
    """

def set_dump_io_flag(dump_io_flag):
    """ Dump input date and output date.

    Parameters
    ----------
    dump_io_flag : bool
```

(下页继续)

(续上页)

```
        if dump_io_flag is true, dump input date and output date, Otherwise not dump.
    """

def set_decoder_env(env_name, env_value):
    """ Set Decoder environment, must set befor Decoder Constructor, else use default values

    Parameters
    ----------
    env_name: str
        Environment name,
        name list: refcounted_frames, extra_frame_buffer_num, rtsp_transport, stimeout, \
            rtsp_flags, buffer_size, max_delay, probesize, analyzeduration.
    env_value: str
        Environment value.
    """
```

3.3.2 Data type

```
# Data type for float32
sail.Dtype.BM_FLOAT32
# Data type for int8
sail.Dtype.BM_INT8
# Data type for uint8
sail.Dtype.BM_UINT8
# Data type for int32
sail.Dtype.BM_INT32
# Data type for uint32
sail.Dtype.BM_UINT32
```

3.3.3 PaddingAtrr

```
def __init__():
    """ Constructor with no parameters. """

def __init__(stx, sty, width, height, r, g, b):
    """ Constructor PaddingAtrr.

    Parameters
    ----------
    stx : int
        Offset x information relative to the origin of dst image
    sty : int
        Offset y information relative to the origin of dst image
    width : int
        The width after resize
    height : int
        The height after resize
```

(下页继续)

(续上页)

```
    r : int
        Pixel value information of R channel
    g : int
        Pixel value information of G channel
    b : int
        Pixel value information of B channel
    """

def set_stx(stx):
    """ set offset stx.

    Parameters
    ----------
    stx : int
        Offset x information relative to the origin of dst image
    """

def set_sty(sty):
    """ set offset sty.

    Parameters
    ----------
    sty : int
        Offset y information relative to the origin of dst image
    """

def set_w(width):
    """ set widht.

    Parameters
    ----------
    width : int
        The width after resize
    """

def set_h(height):
    """ set height.

    Parameters
    ----------
    height : int
        The height after resize
    """

def set_r(r):
    """ set R.

    Parameters
    ----------
    r : int
        Pixel value information of R channel
```

(下页继续)

(续上页)

```
    """

def set_g(g):
    """ set G.

    Parameters
    ----------
    g : int
        Pixel value information of G channel
    """

def set_g(b):
    """ set B.

    Parameters
    ----------
    b : int
        Pixel value information of B channel
    """
```

3.3.4 sail.Handle

```
def __init__(tpu_id):
    """ Constructor handle instance

    Parameters
    ----------
    tpu_id : int
        create handle with tpu Id
    """

def get_device_id():
    """ Get tpu id of this handle.

    Returns
    -------
    tpu_id : int
        tpu id of this handle.
    """

def get_sn():
    """ Get serial number of this handle.

    Returns
    -------
    serial_number : str
        serial number of this handle.
    """
```

3.3.5 sail.IOMode

```
# Input tensors are in system memory while output tensors are in device memory
sail.IOMode.SYSI
# Input tensors are in device memory while output tensors are in system memory.
sail.IOMode.SYSO
# Both input and output tensors are in system memory.
sail.IOMode.SYSIO
# Both input and output tensors are in device memory.
sail.IOMode.DEVIO
```

3.3.6 sail.Tensor

1). Tensor

```
def __init__(handle, data, own_sys_data=True):
    """ Constructor allocates device memory of the tensor.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    array_data : numpy.array
        Tensor ndarray data, dtype can be np.float32, np.int8 or np.uint8
    own_sys_data : bool, default: True
        Indicator of whether own system memory, If false, the memory will be copied to
↪device directly
    """

def __init__(handle, shape, dtype, own_sys_data, own_dev_data):
    """ Constructor allocates system memory and device memory of the tensor.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    shape : tuple
        Tensor shape
    dytpe : sail.Dtype
        Data type
    own_sys_data : bool
        Indicator of whether own system memory
    own_dev_data : bool
        Indicator of whether own device memory
    """
```

2). shape

```
def shape():
    """ Get shape of the tensor.
```

(下页继续)

(续上页)

```
    Returns
    -------
    tensor_shape : list
        Shape of the tensor
    """
```

3). asnumpy

```
def asnumpy():
    """ Get system data of the tensor.

    Returns
    -------
    data : numpy.array
        System data of the tensor, dtype can be np.float32, np.int8
        or np.uint8 with respective to the dtype of the tensor.
    """

def asnumpy(shape):
    """ Get system data of the tensor.

    Parameters
    ----------
    shape : tuple
        Tensor shape want to get

    Returns
    -------
    data : numpy.array
        System data of the tensor, dtype can be np.float32, np.int8
        or np.uint8 with respective to the dtype of the tensor.
    """
```

4). update_data

```
def update_data(data):
    """ Update system data of the tensor. The data size should not exceed
        the tensor size, and the tensor shape will not be changed.

    Parameters
    -------
    data : numpy.array
        Data.
    """
```

5). scale_from

```
def scale_from(data, scale):
    """ Scale data to tensor in system memory.
```

(下页继续)

(续上页)

```
    Parameters
    -------
    data : numpy.array with dtype of float32
        Data.
    scale : float32
        Scale value.
    """
```

6). scale_to

```
def scale_to(scale):
    """ Scale tensor to data in system memory.

    Parameters
    -------
    scale : float32
        Scale value.

    Returns
    -------
    data : numpy.array with dtype of float32
        Data.
    """

def scale_to(scale, shape):
    """ Scale tensor to data in system memory.

    Parameters
    -------
    scale : float32
        Scale value.
    shape : tuple
        Tensor shape want to get

    Returns
    -------
    data : numpy.array with dtype of float32
        Data.
    """
```

7). reshape

```
def reshape(shape):
    """ Reset shape of the tensor.

    Parameters
    -------
    shape : list
        New shape of the tensor
    """
```

8). own_sys_data

```
def own_sys_data():
    """ Judge if the tensor owns data pointer in system memory.

    Returns
    -------
    judge_ret : bool
        True for owns data pointer in system memory.
    """
```

9). own_dev_data

```
def own_dev_data():
    """ Judge if the tensor owns data in device memory.

    Returns
    -------
    judge_ret : bool
        True for owns data in device memory.
    """
```

10). sync_s2d

```
def sync_s2d():
    """ Copy data from system memory to device memory.
    """

def sync_s2d(size):
    """ Copy data from system memory to device memory with specified size.

    Parameters
    ----------
    size : int
        Byte size to be copied
    """
```

11). sync_d2s

```
def sync_d2s():
    """ Copy data from device memory to system memory.
    """

def sync_d2s(size):
    """ Copy data from device memory to system memory with specified size.

    Parameters
    ----------
    size : int
        Byte size to be copied
    """
```

3.3.7 sail.Engine

1). Engine

```
def __init__(tpu_id):
    """ Constructor does not load bmodel.

    Parameters
    ----------
    tpu_id : int
        TPU ID. You can use bm-smi to see available IDs
    """

def __init__(handle):
    """ Constructor does not load bmodel.

    Parameters
    ----------
    hanle : Handle
       A Handle instance
    """

def __init__(bmodel_path, tpu_id, mode):
    """ Constructor loads bmodel from file.

    Parameters
    ----------
    bmodel_path : str
        Path to bmodel
    tpu_id : int
        TPU ID. You can use bm-smi to see available IDs
    mode : sail.IOMode
        Specify the input/output tensors are in system memory
        or device memory
    """

def __init__(bmodel_bytes, bmodel_size, tpu_id, mode):
    """ Constructor using default input shapes with bmodel which
    loaded in memory

    Parameters
    ----------
    bmodel_bytes : bytes
        Bytes of  bmodel in system memory
    bmodel_size : int
        Bmodel byte size
    tpu_id : int
        TPU ID. You can use bm-smi to see available IDs
    mode : sail.IOMode
        Specify the input/output tensors are in system memory
        or device memory
    """
```

2). get_handle

```
def get_handle():
    """ Get Handle instance.

    Returns
    -------
    handle: sail.Handle
       Handle instance
    """
```

3). load

```
def load(bmodel_path):
    """ Load bmodel from file.

    Parameters
    ----------
    bmodel_path : str
        Path to bmodel
    """

def load(bmodel_bytes, bmodel_size):
    """ Load bmodel from file.

    Parameters
    ----------
    bmodel_bytes : bytes
        Bytes of  bmodel in system memory
    bmodel_size : int
        Bmodel byte size
    """
```

4). get_graph_names

```
def get_graph_names():
    """ Get all graph names in the loaded bmodels.

    Returns
    -------
    graph_names : list
        Graph names list in loaded context
    """
```

5). set_io_mode

```
def set_io_mode(graph_name, mode):
    """ Set IOMode for a graph.

    Parameters
    ----------
    graph_name: str
```

(下页继续)

(续上页)

```
        The specified graph name
    mode : sail.IOMode
        Specified io mode
    """
```

6). get_input_names

```
def get_input_names(graph_name):
    """ Get all input tensor names of the specified graph.

    Parameters
    ----------
    graph_name : str
        Specified graph name

    Returns
    -------
    input_names : list
        All the input tensor names of the graph
    """
```

7). get_output_names

```
def get_output_names(graph_name):
    """ Get all output tensor names of the specified graph.

    Parameters
    ----------
    graph_name : str
        Specified graph name

    Returns
    -------
    input_names : list
        All the output tensor names of the graph
    """
```

8). get_max_input_shapes

```
def get_max_input_shapes(graph_name):
    """ Get max shapes of input tensors in a graph.
        For static models, the max shape is fixed and it should not be changed.
        For dynamic models, the tensor shape should be smaller than or equal to
        the max shape.

    Parameters
    ----------
    graph_name : str
        The specified graph name
```

(下页继续)

(续上页)

```
    Returns
    -------
    max_shapes : dict {str : list}
        Max shape of the input tensors
    """
```

9). get_input_shape

```
def get_input_shape(graph_name, tensor_name):
    """ Get the maximum dimension shape of an input tensor in a graph.
        There are cases that there are multiple input shapes in one input name,
        This API only returns the maximum dimension one for the memory allocation
        in order to get the best performance.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified input tensor name

    Returns
    -------
    tensor_shape : list
        The maxmim dimension shape of the tensor
    """
```

10). get_max_output_shapes

```
def get_max_output_shapes(graph_name):
    """ Get max shapes of input tensors in a graph.
        For static models, the max shape is fixed and it should not be changed.
        For dynamic models, the tensor shape should be smaller than or equal to
        the max shape.

    Parameters
    ----------
    graph_name : str
        The specified graph name

    Returns
    -------
    max_shapes : dict {str : list}
        Max shape of the output tensors
    """
```

11). get_output_shape

```
def get_output_shape(graph_name, tensor_name):
    """ Get the shape of an output tensor in a graph.
```

(下页继续)

(续上页)

```
    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    tensor_shape : list
        The shape of the tensor
    """
```

12). get_input_dtype

```
def get_input_dtype(graph_name, tensor_name)
    """ Get scale of an input tensor. Only used for int8 models.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    scale: sail.Dtype
        Data type of the input tensor
    """
```

13). get_output_dtype

```
def get_output_dtype(graph_name, tensor_name)
    """ Get scale of an output tensor. Only used for int8 models.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    scale: sail.Dtype
        Data type of the output tensor
    """
```

14). get_input_scale

```
def get_input_scale(graph_name, tensor_name)
    """ Get scale of an input tensor. Only used for int8 models.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    scale: float32
        Scale of the input tensor
    """
```

15). get_output_scale

```
def get_output_scale(graph_name, tensor_name)
    """ Get scale of an output tensor. Only used for int8 models.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    scale: float32
        Scale of the output tensor
    """
```

16). process

```
def process(graph_name, input_tensors):
    """ Inference with provided system data of input tensors.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    input_tensors : dict {str : numpy.array}
        Data of all input tensors in system memory

    Returns
    -------
    output_tensors : dict {str : numpy.array}
        Data of all output tensors in system memory
    """
```

(下页继续)

(续上页)

```
def process(graph_name, input_tensors, output_tensors):
    """ Inference with provided input and output tensors.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    input_tensors : dict {str : sail.Tensor}
        Input tensors managed by user
    output_tensors : dict {str : sail.Tensor}
        Output tensors managed by user
    """

def process(graph_name, input_tensors, input_shapes, output_tensors):
    """ Inference with provided input tensors, input shapes and output tensors.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    input_tensors : dict {str : sail.Tensor}
        Input tensors managed by user
    input_shapes : dict {str : list}
        Shapes of all input tensors
    output_tensors : dict {str : sail.Tensor}
        Output tensors managed by user
    """
```

17). get_device_id

```
def get_device_id():
    """ Get device id of this engine

    Returns
    ----------
    tpu_id : int
        tpu id of this engine

    """
```

**18). create_input_tensors_map **

```
def create_input_tensors_map(graph_name, create_mode):
    """ Create input tensors map, according to and bmodel.

    Parameters:
    ----------
    graph_name : str
        The specified graph name.
```

(下页继续)

(续上页)

```
    create_mode: int
        Tensor Create mode,
        case 0: only allocate system memory;
        case 1: only allocate device memory;
        case other: according to engine IOMode.

    Returns
    -------
    output: dict[str,Tensor]
        Output result.
    """
```

**19). create_output_tensors_map **

```
def create_output_tensors_map(graph_name, create_mode):
    """ Create output tensors map, according to and bmodel.

    Parameters:
    ----------
    graph_name : str
        The specified graph name.
    create_mode: int
        Tensor Create mode,
        case 0: only allocate system memory;
        case 1: only allocate device memory;
        case other: according to engine IOMode.

    Returns
    -------
    output: dict[str,Tensor]
        Output result.
    """
```

3.3.8 sail.MultiEngine

1). MultiEngine

```
def __init__(bmodel_path, device_ids, sys_out, graph_idx):
    """ Constructor load bmodel.

    Parameters
    ----------
    bmodel_path : str
        Path to bmodel
    device_ids : list[int]
        TPU ID. You can use bm-smi to see available IDs
    sys_out : bool, default: True
        The flag of copy result to system memory.
    graph_idx : int, default: 0
```

(下页继续)

(续上页)

```
        The specified graph index
    """
```

**2). set_print_flag **

```
def set_print_flag(print_flag):
    """ Print debug messages.

    Parameters
    ----------
    print_flag : bool
        if print_flag is true, print debug messages
    """
```

**3). set_print_time **

```
def set_print_time(print_flag):
    """ Print main process time use.

    Parameters
    ----------
    print_flag : bool
        if print_flag is true, print main process time use, Otherwise not print.
    """
```

**4). get_device_ids **

```
def get_device_ids():
    """ Get device ids of this MultiEngine.

    Returns
    -------
    device_ids : list[int]
        tpu ids of this MultiEngine.
    """
```

**5). get_graph_names **

```
def get_graph_names()
    """ Get all graph names in the loaded bmodels.

    Returns
    -------
    graph_names : list
        Graph names list in loaded context
    """
```

6). get_input_names

```
def get_input_names(graph_name):
    """ Get all input tensor names of the specified graph.

    Parameters
    ----------
    graph_name : str
        Specified graph name

    Returns
    -------
    input_names : list
        All the input tensor names of the graph
    """
```

7). get_output_names

```
def get_output_names(graph_name):
    """ Get all output tensor names of the specified graph.

    Parameters
    ----------
    graph_name : str
        Specified graph name

    Returns
    -------
    input_names : list
        All the output tensor names of the graph
    """
```

8). get_input_shape

```
def get_input_shape(graph_name, tensor_name):
    """ Get the maximum dimension shape of an input tensor in a graph.
        There are cases that there are multiple input shapes in one input name,
        This API only returns the maximum dimension one for the memory allocation
        in order to get the best performance.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified input tensor name

    Returns
    -------
    tensor_shape : list
        The maxmim dimension shape of the tensor
    """
```

9). get_output_shape

```
def get_output_shape(graph_name, tensor_name):
    """ Get the shape of an output tensor in a graph.

    Parameters
    ----------
    graph_name : str
        The specified graph name
    tensor_name : str
        The specified output tensor name

    Returns
    -------
    tensor_shape : list
        The shape of the tensor
    """
```

10). process

```
def process(input_tensors):
    """ Inference with provided system data of input tensors.

    Parameters
    ----------
    input_tensors : dict {str : numpy.array}
        Data of all input tensors in system memory

    Returns
    -------
    output_tensors : dict {str : numpy.array}
        Data of all output tensors in system memory
    """
```

3.3.9 sail.bm_image

```
def width():
    """ Get width of img.

    Returns
    ----------
    width : int
        width of img
    """

def height():
    """ Get height of img.

    Returns
    ----------
    height : int
```

(下页继续)

(续上页)

```
        height of img
    """

def format():
    """ Get format of img.

    Returns
    ----------
    format : bm_image_format_ext
        format of img
    """

def dtype():
    """ Get dtype of img.

    Returns
    ----------
    dtype : bm_image_data_format_ext
        dtype of img
    """
```

3.3.10 sail.BMImage

1). BMImage

```
def __init__():
    """ Constructor.
    """

def __init__(handle, h, w, format, dtype):
    """ Constructor.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    h: int
        The height of img
    w: int
        The width of img
    format : bm_image_format_ext
        The format of img
    dtype: sail.bm_image_data_format_ext
        The data type of img
    """
```

2). width

```
def width():
    """ Get the img width.

    Returns
    ----------
    width : int
        The width of img
    """
```

3). height

```
def height():
    """ Get the img height.

    Returns
    ----------
    height : int
        The height of img
    """
```

4). format

```
def format():
    """ Get the img format.

    Returns
    ----------
    format : bm_image_format_ext
        The format of img
    """
```

5). dtype

```
def dtype():
    """ Get the img dtype.

    Returns
    ----------
    dtype: bm_image_data_format_ext
        The data type of img
    """
```

6). data

```
def data():
    """ Get inner bm_image.

    Returns
    ----------
    img : bm_image
        the data of img
    """
```

3.3.11 sail.BMImageArray

1). BMImageArray

```
def __init__():
    """ Constructor.
    """

def __init__(handle, h, w, format, dtype):
    """ Constructor.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    h : int
        Height instance
    w : int
        Width instance
    format : bm_image_format_ext
        Format instance
    dtype : bm_image_data_format_ext
        Dtype instance
    """
```

2). __getitem__

```
def __getitem__(i):
    """ Get the bm_image from index i.

    Parameters
    ----------
    i : int
        Index of the specified location.

    Returns
    ----------
    img : sail.bm_image
        result bm_image
    """
```

3). __setitem__

```
def __setitem__(i, data):
    """ Copy the image to the specified index.

    Parameters
    ----------
    i: int
        Index of the specified location.
    data: sail.bm_image
```

(下页继续)

(续上页)

```
    Input image
"""
```

4). copy_from

```
def copy_from(i, data):
    """ Copy the image to the specified index.

    Parameters
    ----------
    i: int
        Index of the specified location.
    data: sail.BMImage
        Input image
    """
```

5). attach_from

```
def attach_from(i, data):
    """  Attach the image to the specified index.(Because there is no memory copy, the␣
↪original data needs to be cached)

    Parameters:
    ----------
    i: int
        Index of the specified location.
    data: BMImage
        Input image.
    """
```

3.3.12 sail.Decoder

1). Decoder

```
def __init__(file_path, compressed=True, tpu_id=0):
    """ Constructor.

    Parameters
    ----------
    file_path : str
      Path or rtsp url to the video/image file
    compressed : bool, default: True
      Whether the format of decoded output is compressed NV12.
    tpu_id: int, default: 0
      ID of TPU, there may be more than one TPU for PCIE mode.
    """
```

2). is_opened

```
def is_opened():
    """ Judge if the source is opened successfully.

    Returns
    ----------
    judge_ret : bool
        True for success and False for failure
    """
```

3). read

```
def read(handle, image):
    """ Read an image from the Decoder.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    image : sail.BMImage
        BMImage instance

    Returns
    ----------
    judge_ret : int
        0 for success and others for failure
    """

def read(handle):
    """ Read an image from the Decoder.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance

    Returns
    ----------
    image : sail.BMImage
        BMImage instance
    """
```

4). read_

```
def read_(handle, image):
    """ Read an image from the Decoder.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    image : sail.bm_image
```

(下页继续)

(续上页)

```
        bm_image instance

    Returns
    ----------
    judge_ret : int
        0 for success and others for failure
    """
```

5). get_frame_shape

```
def get_frame_shape():
    """ Get frame shape in the Decoder.

    Returns
    ----------
    frame_shape : list
        The shape of the frame
    """
```

5). release

```
def release():
    """ Release the Decoder.
    """
```

6). reconnect

```
def reconnect():
    """ Reconnect the Decoder.
    """
```

3.3.13 sail.Bmcv

1). Bmcv

```
def __init__(handle):
    """ Constructor.

    Parameters
    ----------
    handle : sail.Handle
        Handle instance
    """
```

2). bm_image_to_tensor

```
def bm_image_to_tensor(image):
    """ Convert image to tensor.
```

(下页继续)

(续上页)

```
    Parameters
    ----------
    image : sail.BMImage | sail.BMImageArray
        BMImage/BMImageArray instance

    Returns
    -------
    tensor : sail.Tensor
        Tensor instance
    """

def bm_image_to_tensor(image, tensor):
    """ Convert image to tensor.

    Parameters
    ----------
    image : sail.BMImage | sail.BMImageArray
        BMImage/BMImageArray instance

    tensor : sail.Tensor
        Tensor instance
    """
```

3). tensor_to_bm_image

```
def tensor_to_bm_image(tensor, bgr2rgb=False):
    """ Convert tensor to image.

    Parameters
    ----------
    tensor : sail.Tensor
        Tensor instance
    bgr2rgb : bool, default: False
        Swap color channel

    Returns
    -------
    image : sail.BMImage
        BMImage instance
    """

def tensor_to_bm_image(tensor, img, bgr2rgb=False):
    """ Convert tensor to image.

    Parameters
    ----------
    tensor : sail.Tensor
        Tensor instance
    img : sail.BMImage | sail.BMImageArray
```

(下页继续)

(续上页)

```
        BMImage/BMImageArray instance
    bgr2rgb : bool, default: False
        Swap color channel

    Returns
    -------
    image : sail.BMImage
        BMImage instance
    """
```

4). crop_and_resize

```
def crop_and_resize(input, crop_x0, crop_y0, crop_w, crop_h, resize_w, resize_h):
    """ Crop then resize an image.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def crop_and_resize(input, crop_x0, crop_y0, crop_w, crop_h, resize_w, resize_h):
    """ Crop then resize an image array.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
```

(下页继续)

(续上页)

```
        Width of the crop window
    crop_h : int
        Height of the crop window
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

5). crop

```
def crop(input, crop_x0, crop_y0, crop_w, crop_h):
    """ Crop an image with given window.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def crop(input, crop_x0, crop_y0, crop_w, crop_h):
    """ Crop an image array with given window.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
```

(下页继续)

(续上页)

```
        Width of the crop window
    crop_h : int
        Height of the crop window

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

6). resize

```
def resize(input, resize_w, resize_h):
    """ Resize an image with interpolation of INTER_NEAREST.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def resize(input, resize_w, resize_h):
    """ Resize an image array with interpolation of INTER_NEAREST.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

7). vpp_crop_and_resize

```
def vpp_crop_and_resize(input, crop_x0, crop_y0, crop_w, crop_h, resize_w,␣
↪resize_h):
    """ Crop then resize an image using vpp.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def vpp_crop_and_resize(input, crop_x0, crop_y0, crop_w, crop_h, resize_w,␣
↪resize_h):
    """ Crop then resize an image array using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImageArray
```

(下页继续)

(续上页)

```
    Output image array
  """
```

8). vpp_crop_and_resize_padding

```
def vpp_crop_and_resize_padding(input, crop_x0, crop_y0, crop_w, crop_h,␣
↪resize_w, resize_h, padding):
  """ Crop then resize an image using vpp.

  Parameters
  ----------
  input : sail.BMImage
    Input image
  crop_x0 : int
    Start point x of the crop window
  crop_y0 : int
    Start point y of the crop window
  crop_w : int
    Width of the crop window
  crop_h : int
    Height of the crop window
  resize_w : int
    Target width
  resize_h : int
    Target height
  padding : PaddingAtrr
    padding info

  Returns
  ----------
  output : sail.BMImage
    Output image
  """

def vpp_crop_and_resize_padding(input, crop_x0, crop_y0, crop_w, crop_h,␣
↪resize_w, resize_h, padding):
  """ Crop then resize an image array using vpp.

  Parameters
  ----------
  input : sail.BMImageArray
    Input image array
  crop_x0 : int
    Start point x of the crop window
  crop_y0 : int
    Start point y of the crop window
  crop_w : int
    Width of the crop window
  crop_h : int
    Height of the crop window
```

(下页继续)

(续上页)

```
    resize_w : int
        Target width
    resize_h : int
        Target height
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

9). vpp_crop

```
def vpp_crop(input, crop_x0, crop_y0, crop_w, crop_h):
    """ Crop an image with given window using vpp.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def vpp_crop(input, crop_x0, crop_y0, crop_w, crop_h):
    """ Crop an image array with given window using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
```

(下页继续)

(续上页)

```
    crop_h : int
        Height of the crop window

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

10). vpp_crop_padding

```
def vpp_crop_padding(input, crop_x0, crop_y0, crop_w, crop_h, padding):
    """ Crop an image with given window using vpp.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def vpp_crop_padding(input, crop_x0, crop_y0, crop_w, crop_h, padding):
    """ Crop an image array with given window using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
```

(下页继续)

(续上页)

```
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

11). vpp_resize

```
def vpp_resize(input, resize_w, resize_h):
    """ Resize an image with interpolation of INTER_NEAREST using vpp.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def vpp_resize(input, resize_w, resize_h):
    """ Resize an image array with interpolation of INTER_NEAREST using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    resize_w : int
        Target width
    resize_h : int
        Target height

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """

 def vpp_resize(input, output, resize_w, resize_h):
    """ Resize an image with interpolation of INTER_NEAREST using vpp.
```

(下页继续)

(续上页)

```
    Parameters
    ----------
    input : sail.BMImage
        Input image
    output : sail.BMImage
        Output image
    resize_w : int
        Target width
    resize_h : int
        Target height
    """

def vpp_resize(input, output, resize_w, resize_h):
    """ Resize an image array with interpolation of INTER_NEAREST using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    output : sail.BMImageArray
        Output image array
    resize_w : int
        Target width
    resize_h : int
        Target height
    """
```

12). vpp_resize_padding

```
def vpp_resize_padding(input, resize_w, resize_h, padding):
    """ Resize an image with interpolation of INTER_NEAREST using vpp.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    resize_w : int
        Target width
    resize_h : int
        Target height
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def vpp_resize_padding(input, resize_w, resize_h, padding):
```

(下页继续)

(续上页)

```
    """ Resize an image array with interpolation of INTER_NEAREST using vpp.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    resize_w : int
        Target width
    resize_h : int
        Target height
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

13). warp

```
def warp(input, matrix):
    """ Applies an affine transformation to an image.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    matrix: 2d list
        2x3 transformation matrix

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def warp(input, matrix):
    """ Applies an affine transformation to an image array.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    matrix: 2d list
        2x3 transformation matrix

    Returns
    ----------
    output : sail.BMImageArray
```

(下页继续)

(续上页)

```
        Output image array
    """
```

14). convert_to

```
def convert_to(input, alpha_beta):
    """ Applies a linear transformation to an image.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    alpha_beta: tuple
        (a0, b0), (a1, b1), (a2, b2) factors

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def convert_to(input, alpha_beta):
    """ Applies a linear transformation to an image array.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    alpha_beta: tuple
        (a0, b0), (a1, b1), (a2, b2) factors

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """

def convert_to(input, output, alpha_beta):
    """ Applies a linear transformation to an image.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    output : sail.BMImage
        Output image
    alpha_beta: tuple
        (a0, b0), (a1, b1), (a2, b2) factors
    """
```

(下页继续)

(续上页)

```
def convert_to(input, output, alpha_beta):
    """ Applies a linear transformation to an image array.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array
    alpha_beta: tuple
        (a0, b0), (a1, b1), (a2, b2) factors
    output : sail.BMImageArray
        Output image array
    """
```

15). yuv2bgr

```
def yuv2bgr(input):
    """ Convert an image from YUV to BGR.

    Parameters
    ----------
    input : sail.BMImage
        Input image

    Returns
    ----------
    output : sail.BMImage
        Output image
    """

def yuv2bgr(input):
    """ Convert an image array from YUV to BGR.

    Parameters
    ----------
    input : sail.BMImageArray
        Input image array

    Returns
    ----------
    output : sail.BMImageArray
        Output image array
    """
```

16). rectangle

```
def rectangle(image, x0, y0, w, h, color, thickness=1):
    """ Draw a rectangle on input image.

    Parameters
    ----------
```

(下页继续)

(续上页)

```
    image : sail.BMImage
        Input image
    x0 : int
        Start point x of rectangle
    y0 : int
        Start point y of rectangle
    w : int
        Width of rectangle
    h : int
        Height of rectangle
    color : tuple
        Color of rectangle
    thickness : int
        Thickness of rectangle

    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

17). imwrite

```
def imwrite(file_name, image):
    """ Save the image to the specified file.

    Parameters
    ----------
    file_name : str
        Name of the file
    output : sail.BMImage
        Image to be saved

    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

18). get_handle

```
def get_handle():
    """ Get Handle instance.

    Returns
    -------
    handle: sail.Handle
       Handle instance
"""
```

19). crop_and_resize_padding

```
def crop_and_resize_padding(input, crop_x0, crop_y0, crop_w, crop_h, resize_w,␣
↪resize_h, padding):
    """ Crop then resize an image.

    Parameters
    ----------
    input : sail.BMImage
        Input image
    crop_x0 : int
        Start point x of the crop window
    crop_y0 : int
        Start point y of the crop window
    crop_w : int
        Width of the crop window
    crop_h : int
        Height of the crop window
    resize_w : int
        Target width
    resize_h : int
        Target height
    padding : PaddingAtrr
        padding info

    Returns
    ----------
    output : sail.BMImage
        Output image
    """
```

20). rectangle_

```
def rectangle_(image, x0, y0, w, h, color, thickness=1):
    """ Draw a rectangle on input image.

    Parameters
    ----------
    image : sail.bm_image
        Input image
    x0 : int
        Start point x of rectangle
    y0 : int
        Start point y of rectangle
    w : int
        Width of rectangle
    h : int
        Height of rectangle
    color : tuple
        Color of rectangle
    thickness : int
        Thickness of rectangle
```

(下页继续)

(续上页)

```
    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

21). imwrite_

```
def imwrite_(file_name, image):
    """ Save the image to the specified file.

    Parameters
    ----------
    file_name : str
        Name of the file
    output : sail.bm_image
        Image to be saved

    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

22). convert_format

```
def convert_format(input, output):
    """Convert input to output format.

    Parameters
    ----------
    input : sail.BMImage
        BMimage instance
    output : sail.BMImage
        output image
    """

def convert_format(input):
    """Convert an image to BGR PLANAR format.

    Parameters
    ----------
    input : sail.BMImage
        BMimage instance

    Returns
    ----------
    output : sail.BMImage
        output image
    """
```

23). vpp_convert_format

```
def vpp_convert_format(input, output):
    """Convert input to output format using vpp.

    Parameters
    ----------
    input : sail.BMImage
        BMimage instance
    output : sail.BMImage
        output image
    """

def vpp_convert_format(input):
    """Convert an image to BGR PLANAR format using vpp.

    Parameters
    ----------
    input : sail.BMImage
        BMimage instance

    Returns
    ----------
    output : sail.BMImage
        output image
    """
```

**24). putText **

```
def putText(input, text, x, y, color, fontScale, thickness):
    """ Draws a text on the image

    Parameters
    ----------
    input : sail.BMImage
        BMimage instance
    text: str
        Text to write on an image.
    x: int
        Start point x
    y: int
        Start point y
    color : tuple
        Color of text
    thickness : int
        Thickness of text

    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

**25). putText_ **

```
def putText_(input, text, x, y, color, fontScale, thickness):
    """ Draws a text on the image

    Parameters
    ----------
    input : sail.bm_image
        bm_image instance
    text: str
        Text to write on an image.
    x: int
        Start point x
    y: int
        Start point y
    color : tuple
        Color of text
    thickness : int
        Thickness of text

    Returns
    ----------
    process_status : int
        0 for success and others for failure
    """
```

**26). image_add_weighted **

```
def image_add_weighted(input0, alpha, input1, beta, gamma, output):
    """ Calculates the weighted sum of two images

    Parameters
    ----------
    input0 : sail.BMImage
        BMimage instance.
    alpha : float
        alpha instance.
    input1 : sail.BMImage
        BMImage instance.
    beta: float
        beta instance.
    gamma: float
        gamma instance.
    output: BMImage
        result BMImage, output = input1 * alpha + input2 * beta + gamma.
    """

def image_add_weighted(input0, alpha, input1, beta, gamma):
    """ Calculates the weighted sum of two images

    Parameters
    ----------
```

(下页继续)

(续上页)

```
    input0 : sail.BMImage
        BMimage instance.
    alpha : float
        alpha instance.
    input1 : sail.BMImage
        BMImage instance.
    beta: float
        beta instance.
    gamma: float
        gamma instance.

    Returns
    -------
    output: BMImage
        result BMImage, output = input1 * alpha + input2 * beta + gamma.
    """
```

**27). image_copy_to **

```
def image_copy_to(self, input, output, start_x, start_y):
    """ Copy the input to the output.

    Parameters:
    ----------
    input: BMImage|BMImageArray
        Input image or image array.
    output: BMImage|BMImageArray
        Output image or image array.
    start_x: int
        Point start x.
    start_y: int
        Point start y.
    """
```

**28). image_copy_to_padding **

```
def image_copy_to_padding(self, input, output, padding_r, padding_g, padding_b, start_
↪x, start_y):
    """ Copy the input to the output width padding.

    Parameters:
    ----------
    input: BMImage|BMImageArray
        Input image or image array.
    output: BMImage|BMImageArray
        Output image or image array.
    padding_r: int
        r value for padding.
    padding_g: int
        g value for padding.
```

(下页继续)

(续上页)

```
    padding_b: int
        b value for padding.
    start_x: int
        point start x.
    start_y: int
        point start y.
    """
```

**28). nms **

```
def nms(input, threshold) :
    """ Do nms use tpu.

    Parameters:
    ----------
    input: float
        input proposal array, shape must be (n,5) n<56000, proposal:[left,top,right,bottom,
↪score].
    threshold: float
        nms threshold.

    Returns:
    ----------
    return nms result, numpy.ndarray[Any, numpy.dtype[numpy.float32]]
    """
```