MaxEngine-Lite-V0.9.9

C API Documentation

Table of Contents

Chapter 1 Overview

1.1 Overview

本文档对 MaxEngine-Lite 的 C API 的说明。包括 C Infer API 和 C Attention API 两部分。针对每部分，分别对数据类型，接口函数等进行了详细的说明。

1.2 C Infer API function list

1) MXInferCreateSimple
2) MXInferCreateSimpleShared
3) MXInferCreate
4) MXInferCreateShared
5) MXInferGetOutputShape
6) MXInferSetInput
7) MXInferGetInputPtr
8) MXInferReshape
9) MXInferForward
10) MXInferGetOutput
11) MXInferGetOutputPtr
12) MXInferGetOutputNum
13) MXInferGetOutputName
14) MXInferFree

1.3 C Attention API function list

1) MXAttentionCreateSimple
2) MXAttentionCreate
3) MXAttentionCreatePartialOut
4) MXAttentionSetInput
5) MXAttentionReshape
6) MXAttentionForward
7) MXAttentionGetOutputShape
8) MXAttentionGetOutput
9) MXAttentionFree

Chapter 2 C API Datetypes Reference

2.1 mx_uint

mx_uint : 对“unsigned int”类型的重新命名；
typedef unsigned int mx_uint;

2.2 mx_float

mx_float : 对“float”类型的重新命名；
typedef float mx_float;

2.3 ModelHandle

ModelHandle : predictor 的手柄(For Infer API)
typedef void *ModelHandle;

2.4 AttentionHandle

AttentionHandle: attention engine 的手柄(For Attention API)
typedef void * AttentionHandle;

2.4 DEV_TYPE

DEV_TYPE：对不同设备类型，使用枚举变量定义，调用函数时“dev_type”的位置可以直接使用 MAXENGINE_X 标识不同设备 (For Infer API)。

```
enum DEV_TYPE {
    MAXENGINE_CPU = 1,              //表示一般的 x86 的 CPU（PC 和服务器端）
    MAXENGINE_GPU,                  //表示一般的NVIDIA 的设备（PC 和服务器）
    MAXENGINE_MLU = 6,              //表示寒武纪的智能芯片
    MAXENGINE_ARM_CPU = 8,          //表示移动端的 CPU
    MAXENGINE_ARM_GPU               //表示移动端的 GPU
};
```

Chapter 3 C Infer API Reference

本章节主要介绍 MaxEngine-Lite C 端推理接口函数，对每个函数将会从函数定义、函数功能、参数以及返回值四个方面进行详细说明。

3.1 MXInferCreateSimple

3.1.1 函数的完整定义

```
int MXInferCreateSimple (const char* json_files,
                         const void* param_files,
                         int dev_type, int dev_id,
                         mx_uint num_input_nodes,
                         const char** input_keys,
                         const mx_uint* input_shape_indptr,
                         const mx_uint* input_shape_data,
                         int max_len, int model_type,
                         ModelHandle* out);
```

3.1.2 函数功能

该函数根据输入.json 和.params 的**路径**，实现创建 predictor 的功能。

3.1.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_files	const char*	input	输入模型的.json 文件的路径
2	param_files	const void*	input	输入模型的参数文件.params 的路径
3	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
4	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；
5	num_input_nodes	mx_uint	input	模型的输入结点个数
6	input_keys	const char**	input	模型的所有输入结点的名称

7	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
8	input_shape_data	const mx_uint*	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；
9	max_len	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
10	model_type	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
11	out	ModelHandle*	output	创建的 predictor 的句柄

3.1.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.2 MXInferCreateSimpleShared

3.2.1 函数的完整定义

```
int MXInferCreateSimpleShared(const char* json_files,
                              const void* param_files,
                              int dev_type, int dev_id,
                              mx_uint num_input_nodes,
                              const char** input_keys,
                              const mx_uint* input_shape_indptr,
                              const mx_uint* input_shape_data,
                              int max_len, int model_type,
                              ModelHandle* out, ModelHandle shared));
```

3.2.2 函数功能

该函数根据输入.json 和.params **路径**，实现创建 predictor 的功能，与 3.1 相比增加了共享句柄参数，为了共享 handle 之间的存储，后续可能会修改！

3.2.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_files	const char*	input	输入模型的.json 文件的路径
2	param_files	const void*	input	输入模型的参数文件.params 的路径
3	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
4	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；
5	num_input_nodes	mx_uint	input	模型的输入结点个数
6	input_keys	const char**	input	模型的所有输入结点的名称
7	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
8	input_shape_data	const mx_uint*	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；
9	max_len	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除； 目前设置为 0，表示当前模型；
10	model_type	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除； 目前设置为 0，表示当前模型；
11	out	ModelHandle*	output	创建的 predictor 的句柄
12	shared	ModelHandle	output	创建的共享存储的句柄

3.2.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.3 MXInferCreate

3.3.1 函数的完整定义

int ***MXInferCreate***(const char* json_buffers,
const void* param_buffers,
int param_size,
int dev_type, int dev_id,
mx_uint num_input_nodes,
const char** input_keys,
const mx_uint* input_shape_indptr,
const mx_uint* input_shape_data,
int max_len, int model_type,
ModelHandle* out);

3.3.2 函数功能

该函数根据输入.json 和.params 的实际模型和参数**数据（由路径获取到的数据（buffer））**，实现创建 predictor 的功能。

3.3.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_buffers	const char*	input	输入模型的 json 数据流
2	param_buffers	const void*	input	输入模型的参数 params 的数据流
3	param_size	int	input	输入的模型的参数的数据流的长度（大小）
4	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
5	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；
6	num_input_nodes	mx_uint	input	模型的输入结点个数
7	input_keys	const char**	input	模型的所有输入结点的名称
8	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后

				还需要设置最后 shape 的最后一维的索引）
9	input_shape_data	const mx_uint*	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；
10	max_len	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
11	model_type	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
12	out	ModelHandle*	output	创建的 predictor 的句柄

3.3.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.4 MXInferCreateShared

3.4.1 函数的完整定义

```
int MXInferCreateShared (const char* json_buffers,
                         const void* param_buffers,
                         int param_size,
                         int dev_type, int dev_id,
                         mx_uint num_input_nodes,
                         const char** input_keys,
                         const mx_uint* input_shape_indptr,
                         const mx_uint* input_shape_data,
                         int max_len, int model_type,
                         ModelHandle* out, ModelHandle shared));
```

3.4.2 函数功能

该函数根据输入.json 和.params 实际模型和参数**数据（由路径获取到的数据（buffer））**，实现创建 predictor 的功能，与 3.3 相比增加了共享句柄参数，为了共享 handle 之间的存储，后续可能会修改！

3.4.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_buffers	const char*	input	输入模型的 json 数据流
2	param_buffers	const void*	input	输入模型的参数 params 的

				数据流
3	param_size	int	input	输入的模型的参数的数据流的长度（大小）
4	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
5	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；
6	num_input_nodes	mx_uint	input	模型的输入结点个数
7	input_keys	const char**	input	模型的所有输入结点的名称
8	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
9	input_shape_data	const mx_uint*	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；
10	max_len	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
11	model_type	int	input	该参数为 RNN 等循环网络设置，目前版本 Lite 未使用，后续版本将会移除；目前设置为 0，表示当前模型；
12	out	ModelHandle*	output	创建的 predictor 的句柄
13	shared	ModelHandle	output	创建的共享存储的句柄

3.4.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.5 MXInferGetOutputShape

3.5.1 函数的完整定义

```
int MXInferGetOutputShape (ModelHandle handle,
                           mx_uint index,
                           mx_uint** shape_data,
                           mx_uint* shape_ndim);
```

3.5.2 函数功能

该函数根据创建的 predictor 句柄获取输出形状（shape）的大小。

3.5.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄
2	index	mx_uint	input	想要获取的输出结点的索引值，起始值为 0，如果只有一个输出，则设其为 0。
3	shape_data	mx_uint**	output	获取的相应输出结点的形状（shape 值）
4	shape_ndim	mx_uint*	output	获取的相应输出结点的形状维度

3.5.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.6 MXInferSetInput

3.6.1 函数的完整定义

```
int MXInferSetInput (ModelHandle handle,
                     const char* key,
                     const mx_float* data,
                     mx_uint size,
                     int dev_type,
                     int dev_id);
```

3.6.2 函数功能

该函数实现为创建的 predictor 填充输入值的功能。

3.6.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input、output	创建的 predictor 的句柄
2	key	const char*	input	将要填充的输入数据中各

				个输入结点的名字
3	data	const mx_float*	input	将要填充的输入数据流，要与 MXInferCreate 创建的 shape 大小一致。
4	size	mx_uint	input	将要填充的输入数据流的大小，用于安全检查。
5	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
6	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；

3.6.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.7 MXInferGetInputPtr

3.7.1 函数的完整定义

```
mx_float* MXInferGetInputPtr (ModelHandle handle,
                              const char* key);
```

3.7.2 函数功能

该函数实现：获取指向指定输入结点（key）对应的输入数据值的指针。

3.7.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input、output	创建的 predictor 的句柄
2	key	const char*	input	需要获取的指定结点的名字。

3.7.4 函数返回值

函数返回指向指定输入结点对应的输入数据值的指针。

3.8 MXInferReshape

3.8.1 函数的完整定义

```
int MXInferReshape (ModelHandle handle,
```

```
                const char** input_keys,
                int num_input,
                const mx_uint* input_shape_indptr,
                const mx_uint* input_shape_data);
```

3.8.2 函数功能

该函数实现：对创建的 predictor 的输入 data 进行 reshape 操作。

3.8.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input、output	创建的 predictor 的句柄
2	input_keys	const char**	input	模型的所有输入结点的名称
3	num_input	int	input	输入结点的个数
4	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
5	input_shape_data	const mx_uint**	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；

3.8.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.9 MXInferForward

3.9.1 函数的完整定义

```
int MXInferForward (ModelHandle handle);
```

3.9.2 函数功能

该函数实现：运行一个网络的 forward 过程用于获得网络的输出。

3.9.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input、output	创建的 predictor 的句柄

3.9.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.10 MXInferGetOutput

3.10.1 函数的完整定义

```
int MXInferGetOutput (ModelHandle handle,
                      mx_uint index,
                      mx_float* data,
                      mx_uint size,
                      int dev_type,
                      int dev_id);
```

3.10.2 函数功能

该函数实现：获取网络预测的输出值。

3.10.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄
2	index	mx_uint	input	想要获取的输出结点的索引值，起始值为 0，如果只有一个输出，则设其为 0。
3	data	mx_float*	output	用户创建的存储输出数据的数组。
4	size	mx_uint	input	获取的输出数据的大小，用于安全检查。
5	dev_type	int	input	设备类型（可以直接用 MAXENGINE_CPU ，MAXENGINE_GPU 等赋值（参照 DEV_TYPE 变量））
6	dev_id	int	input	设备卡的 ID 号，如果 dev_type=MAXENGINE_CPU，则其为 0；如果 dev_type=MAXENGINE_GPU，则其表示选择的设备卡的卡号；

3.10.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.11 MXInferGetOutputPtr

3.11.1 函数的完整定义

```
mx_float* MXInferGetOutputPtr (ModelHandle handle,
                               mx_uint index);
```

3.11.2 函数功能

该函数实现：获取指向指定输出结点对应的输出数据的指针。

3.11.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄
2	index	mx_unit	output	需要获取的指定输出结点的序号，如果只有一个输出，则将其置为 0。

3.11.4 函数返回值

返回指向指定输出结点对应的输出数据的指针。

3.12 MXInferGetOutputNum

3.12.1 函数的完整定义

```
int MXInferGetOutputNum (ModelHandle handle,
                         mx_uint *number);
```

3.12.2 函数功能

该函数用于获取输出结点的个数。

3.12.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄
2	number	mx_unit*	output	用于存储获取的输出结点个数的变量。

3.12.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.13 MXInferGetOutputName

3.13.1 函数的完整定义

```
int MXInferGetOutputName (ModelHandle handle,
                          mx_uint out_index,
                          const char** name);
```

3.13.2 函数功能

该函数实现：获取指定输出结点的名字。

3.13.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄
2	index	mx_unit	input	想要获取的输出结点的序号，如果只有一个输出结点，则将其置为 0.
3	name	const char*	output	求得的指定输出结点的名字。

3.13.4 函数返回值

运行成功返回值为 0，失败则返回-1。

3.14 MXInferFree

3.14.1 函数的完整定义

```
int MXInferFree (ModelHandle handle);
```

3.14.2 函数功能

该函数实现：释放创建的 predictor 的句柄。

3.14.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	ModelHandle	input	创建的 predictor 的句柄

3.14.4 函数返回值

运行成功返回值为 0，失败则返回-1。

Chapter 4 C Attention API Reference

本章节主要介绍 MaxEngine-Lite C 端 Attention 接口函数，对每个函数将会从函数定义、函数功能、参数以及返回值四个方面进行详细说明。

4.1 MXAttentionCreateSimple

4.1.1 函数的完整定义

```
int MXAttentionCreateSimple (const char** json_paths,
                             const char** param_paths,
                             int start_sym, int end_sym,
                             mx_uint max_decode_frame,
                             int dev_type, int dev_id,
                             mx_uint num_model_nodes,
                             mx_uint* num_input_nodes,
                             const char** input_keys,
                             const mx_uint* input_shape_indptr,
                             const mx_uint* input_shape_data,
                             mx_uint beam_size,
                             int engine_type,
                             AttentionHandle* out);
```

4.1.2 函数功能

该函数根据输入.json 和.params 文件的**路径**，实现创建 AttentionEngine 的功能。

4.1.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_paths	const char**	input	输入模型的.json 文件的路径
2	param_paths	const char**	input	输入模型的参数文件.params 的路径
3	start_sym	int	input	字典的起始 symbol 位置
4	end_sym	int	input	字典的截止 symbol 位置
5	max_decode_frame	mx_uint	input	解码的最大帧长度
6	dev_type	int	input	设备类型，1: cpu, 2:gpu
7	dev_id	int	input	引擎的设备卡号
8	num_model_nodes	mx_uint	input	模型的结点个数
9	num_input_nodes	mx_uint	input	所有模型的输入结点个数
10	input_keys	const char**	input	所有模型的所有输入结点的名称

11	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值(前后两个值的差值表示形状的长度,因此最后还需要设置最后 shape 的最后一维的索引)
12	input_shape_data	const mx_uint*	input	所有模型的输入结点的形状大小,要与输入结点的名称对应;
13	beam_size	mx_uint	input	选择 beam search 时的大小
14	engine_type	int	input	创建的 engine 的类型
15	out	AttentionHandle*	output	创建的 AttentionEngine 的句柄

4.1.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.2 MXAttentionCreate

4.2.1 函数的完整定义

```
int MXAttentionCreate (const char** json_buffer,
                       const char** param_buffer,
                       mx_uint param_size,
                       int start_sym, int end_sym,
                       mx_uint max_decode_frame,
                       int dev_type, int dev_id,
                       mx_uint num_model_nodes,
                       mx_uint* num_input_nodes,
                       const char** input_keys,
                       const mx_uint* input_shape_indptr,
                       const mx_uint* input_shape_data,
                       mx_uint beam_size,
                       int engine_type,
                       AttentionHandle* out);
```

4.2.2 函数功能

该函数根据输入.json 和.params 的实际模型和参数**数据(由路径获取到的数据(buffer))**，实现创建 AttentionEngine 的功能。

4.2.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_buffer	const char**	input	输入模型的 json 数据流
2	param_files	const char**	input	输入模型的参数 params 的

				数据流
3	param_size	mx_uint	input	输入的模型的参数的数据流的长度（大小）
4	start_sym	int	input	字典的起始 symbol 位置
5	end_sym	int	input	字典的截止 symbol 位置
6	max_decode_frame	mx_uint	input	解码的最大帧长度
7	dev_type	int	input	设备类型，1: cpu, 2:gpu
8	dev_id	int	input	引擎的设备卡号
9	num_model_nodes	mx_uint	input	引擎的模型模式个数
10	num_input_nodes	mx_uint	input	所有模型的输入结点个数
11	input_keys	const char**	input	所有模型的所有输入结点的名称
12	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
13	input_shape_data	const mx_uint*	input	所有模型的输入结点的形状大小，要与输入结点的名称对应；
14	beam_size	mx_uint	input	选择 beam search 时的大小
15	engine_type	int	input	创建的 engine 的类型
16	out	AttentionHandle*	output	创建的 AttentionEngine 的句柄

4.2.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.3 MXAttentionCreatePartialOut

4.3.1 函数的完整定义

```
int MXAttentionCreatePartialOut (const char** json_buffer,
                                 const char** param_buffer,
                                 mx_uint param_size,
                                 int start_sym, int end_sym,
                                 mx_uint max_decode_frame,
                                 int dev_type, int dev_id,
                                 mx_uint num_model_nodes,
                                 mx_uint* num_input_nodes,
                                 const char** input_keys,
                                 const mx_uint* input_shape_indptr,
```

```
                                const mx_uint* input_shape_data,
                                mx_uint* num_output_nodes,
                                const char** output_keys,
                                mx_uint beam_size,
                                int engine_type,
                                AttentionHandle* out);
```

4.3.2 函数功能

该函数根据输入.json 和.params 的实际模型和参数**数据（由路径获取到的数据（buffer））**，实现创建 AttentionEngine 的功能，是创建 AttentionEngine 的实际实现函数，一般在 MXAttentionCreate 中调用。

4.3.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	json_buffer	const char**	input	输入模型的 json 数据流
2	param_files	const char**	input	输入模型的参数 params 的数据流
3	param_size	mx_uint	input	输入的模型的参数的数据流的长度（大小）
4	start_sym	int	input	字典的起始 symbol 位置
5	end_sym	int	input	字典的截止 symbol 位置
6	max_decode_frame	mx_uint	input	解码的最大帧长度
7	dev_type	int	input	设备类型，1: cpu, 2:gpu
8	dev_id	int	input	引擎的设备卡号
9	num_model_nodes	mx_uint	input	引擎的模型模式个数
10	num_input_nodes	mx_uint	input	所有模型的输入结点个数
11	input_keys	const char**	input	所有模型的所有输入结点的名称
12	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
13	input_shape_data	const mx_uint*	input	所有模型的输入结点的形状大小，要与输入结点的名称对应；
14	beam_size	mx_uint	input	选择 beam search 时的大小
15	num_output_nodes	mx_uint*	input	模型输出结点的个数
16	output_keys	const char**	input	模型输出结点的名称
17	engine_type	int	input	创建的 engine 的类型
18	out	AttentionHandle*	output	创建的 AttentionEngine 的句柄

4.3.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.4 MXAttentionSetInput

4.4.1 函数的完整定义

```
int MXAttentionSetInput (AttentionHandle handle,
                         const char* key,
                         const mx_float* data,
                         mx_uint size);
```

4.4.2 函数功能

该函数为 encode 模型的 engine 填充输出数据。

4.4.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input、output	创建的 AttentionEngine 的句柄
2	key	const char*	input	将要填充的输入数据中各个输入结点的名字
3	data	const mx_float*	input	将要填充的输入数据流，要与 MXAttentionCreate 创建的 shape 大小一致。
4	size	mx_uint	input	将要填充的输入数据流的大小，用于安全检查。

4.4.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.5 MXAttentionReshape

4.5.1 函数的完整定义

```
int MXAttentionReshape (AttentionHandle handle,
                        const mx_uint max_decode_frame,
                        const char** input_keys,
                        mx_uint num_model,
                        mx_uint* num_input,
                        const mx_uint* input_shape_indptr,
                        const mx_uint* input_shape_data);
```

4.5.2 函数功能

该函数实现：对创建的 AttentionEngine 的输入 data 进行 reshape 操作。

4.5.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input、output	创建的 AttentionEngine 的句柄
2	max_decode_frame	mx_uint	input	解码的最大帧长度
3	input_keys	const char**	input	模型的所有输入结点的名称
4	num_model	mx_uint	input	引擎中模型模式的数量
5	num_input	mx_uint*	input	模型输入结点的个数
6	input_shape_indptr	const mx_uint*	input	模型的所有输入结点的形状在 input_shape_data 中的索引值（前后两个值的差值表示形状的长度，因此最后还需要设置最后 shape 的最后一维的索引）
7	input_shape_data	const mx_uint**	input	模型的所有输入结点的形状大小，要与输入结点的名称对应；

4.5.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.6 MXAttentionForward

4.6.1 函数的完整定义

int *MXAttentionForward* (AttentionHandle handle);

4.6.2 函数功能

该函数实现：运行网络的 forward 过程用于获得网络的输出。

4.6.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input、output	创建的 AttentionEngine 的句柄

4.6.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.7 MXAttentionGetOutputShape

4.7.1 函数的完整定义

int *MXAttentionGetOutputShape* (AttentionHandle handle,

```
                                    mx_uint index,
                                    int ibatch,
                                    mx_uint** shape_data,
                                    mx_uint* shape_ndim);
```

4.7.2 函数功能

该函数根据创建的 AttentionEngine 句柄获取输出形状（shape）的大小。

4.7.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input	创建的 AttentionEngine 的句柄
2	index	mx_uint	input	想要获取的输出结点的索引值，起始值为 0，如果只有一个输出，则设其为 0。
3	ibatch	int	input	选择的输出 batch 数，如果获取所有 batches 则设置其为-1.
4	shape_data	mx_uint**	output	用于存储 shape 数据的指针
5	shape_ndim	mx_uint*	output	用于存储 shape 维度的值

4.7.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.8 MXAttentionGetOutput

4.8.1 函数的完整定义

```
int MXAttentionGetOutput (AttentionHandle handle,
                                    mx_uint index,
                                    int ibatch,
                                    mx_float* data,
                                    mx_uint size);
```

4.8.2 函数功能

该函数实现：获取 AttentionEngine 的输出。

4.8.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input	创建的 AttentionEngine 的句柄。
2	index	mx_uint	input	想要获取的输出结点的索引值，起始值为 0，如果只

				有一个输出，则设其为 0。
3	ibatch	int	input	选择的输出 batch 数，如果获取所有 batches 则设置其为-1。
4	data	mx_float *	output	指向用户申请的存储输出数据值的指针。
5	size	mx_uint	input	输出数据的大小，用于安全检查。

4.8.4 函数返回值

运行成功返回值为 0，失败则返回-1。

4.9 MXAttentionFree

4.9.1 函数的完整定义

```
int MXAttentionFree (AttentionHandle handle);
```

4.9.2 函数功能

该函数实现：释放创建的 AttentionEngine 的句柄。

4.9.3 函数参数

序号	变量名称	变量类型	输入/输出	变量含义
1	handle	AttentionHandle	input	创建的 AttentionEngine 的句柄

4.9.4 函数返回值

运行成功返回值为 0，失败则返回-1。