* 基于Lite的HybridCNN模型产品化

HybridCNN BatchBuilder的功能

HybridCNN资源打包与加载

 HybirdCNN一致性对比工作

* EDGen引擎集成Lite组件，实现音频帧输入与后验概率输出，完成dec模块的解码功能

MaxEngine-Lite引擎中beam search的适配工作

EDGen引擎Beam search功能适配

EDGen引擎初始化工作

1. MaxEngine-Lite功能设计

* HybridCNNBatchBuilder

class HybridCNNBatchBuilder : public MxBatchBuilderBaseImpl

{

public:

HybridCNNBatchBuilder (const mx\_char\* p\_name, const mx\_char\*\* pp\_keys,

const mx\_char\*\* pp\_values, mx\_uint32 n\_num, DeviceInfo& dev\_info);

mx\_int32 start(mx\_uint32 inst\_id) override;

mx\_int32 stop(mx\_uint32 inst\_id) override;

mx\_void set\_max\_output\_info(const DataInfo\* p\_shape, mx\_uint32 n\_num) override;

mx\_int32 push\_task(mx\_int32 n\_inst\_id,mx\_int32 n\_task\_id, const Datum\* pdata) override;

**//输入外部数据，首窗需要根据计算窗的尺度进行数据的padding操作**

mx\_int32 init() override;

mx\_int32 flush(mx\_int32 n\_inst\_id, mx\_int32 n\_task\_id) override;

**// flush数据，通知指定inst会话，数据已发送完成**

mx\_void InternalThreadEntry() override;

mx\_int32 uninit() override;

mx\_bool get\_is\_input\_full(std::string input\_name, mx\_int32 n\_instid, mx\_int32 n\_task\_num = 0);

mx\_bool has\_task\_ready();

mx\_bool is\_task\_ready(mx\_uint32 inst\_id);

InstNumInfo data\_collect(std::vector<mx\_void\*> data\_ptrs, GraphContextInst\* inst\_out,mx\_int32 max\_task\_out);

**//数据下发处理函数，最复杂，需要根据计算窗的大小从缓存区进行数据获取，并对数据帧的有效性进行设置**

private:

typedef std::unique\_lock<std::mutex> scoped\_lock;

std::mutex mutex\_;

std::vector<mx\_char\*> v\_cpu\_output\_buff\_;

DeviceBuffMgr\* p\_buff\_mgr\_;

mx\_int32 n\_max\_task\_out\_ = 0;

std::unordered\_set<int32\_t> s\_inst\_flushed\_;

std::unordered\_set<mx\_int32> s\_push\_inst\_count\_;

std::unordered\_set<mx\_int32> s\_ready\_instd\_;

std::deque<int32\_t> d\_inst\_id\_ready\_;

std::deque<InstDatum> \* data\_in\_ques\_;

std::vector<mx\_bool> v\_first\_batch\_;

std::condition\_variable condition\_push\_no\_full\_;

std::condition\_variable condition\_data\_;

HybridCNNParam params\_;

* forward

forward函数，数据完成前向计算后，得到网络的后验概率输出。在数据输入的时候，对于每个任务都有信号量进行绑定，因此所有输出的数据都需要通过信号量来唤醒外部等待数据的线程，这个需要根数据的graphContext进行处理，无效的数据帧不要通知，有效的数据帧需要，这个信号量是通过inst\_id和task\_id进行管理的

1. EDGen引擎模块功能设计

* w\_res模块 新增ResLoader\_LITE\_HybridCNN资源类，负责HybridCNN资源的加载和释放，**实际资源的加载函数在HybridCNNPredictor类中**（单例类）
* w\_rec模块，基本保持不动
* w\_fea模块，基本保持不动
* w\_lite\_mlp模块(w\_lite\_mlp\_cnn.cpp)

HybridCNNPredictor单体类

class MXHybridCNNPredictor

{

public:

MAKE\_SINGLETON(MXHybridCNNPredictor)

// inst

int initialize(void\* reserve); **//初始化、逆初始化**

int uninitialize(void\* reserve);

int resAdd(void\* pRes); **//资源加载、释放**

int resDelete();

int creatInst(**DecInst\* wDecInst**, mlpflow::MXInst\* &pInst); **//创建、销毁inst**

int destroyInst(mlpflow::MXInst\* pMXInst);

//session

int start(mlpflow::MXInst\* pMXInst); **//会话start**

**// 启动解码线程**

int stop(mlpflow::MXInst\* pMXInst); **//会话stop**

int reset(mlpflow::MXInst\* pMXInst); **//会话reset**

int waitComplete(mlpflow::MXInst\* pMXInst);**//回话wait**

// task

int pushTask(mlpflow::MXInst\* pMXInst, const float\* data, int size); **//送音频**

**//根据数据生成task，包括信号量、后验输出buffer，并置入任务队列，用于异步解码**

int flushTask(mlpflow::MXInst\* pMXInst, int language\_type); **//送音频结束**

private:

int beamSearch();

**// 解码线程，从任务队列中不断获取后验进行解码**

**// 当任务队列为空时，需要pushTask进行唤醒**

int parseParam(HybridCNNMemRes\* pRes); **//资源解析**

private:

DISALLOW\_COPY\_AND\_ASSIGN(MXHybridCNNPredictor);

private:

std::map<mlpflow::MXInst\*, pMXInstHybridCNNInfo> MXInstToInfoMap\_;**//inst管理**

boost::shared\_mutex inst\_share\_mutex\_;

**// decInst related shared mutex use for creatInst destroyInst**

**// start stop reset and waitComplete**

FixElementMemPool<float>\* frameFeatureMemPool\_; **//音频帧内存池**

FixElementMemPool<float\*> DNNFeatureMemPool\_; **//后验输出内存池**

Shape inputShape\_;

HybridCNNMemRes\* pModelRes\_; **//资源**

typedef struct TagCombineTask

{

float \*pOutFea; **//后验buffer**

IOutputSemaphore\* pLiteSig; **// 同步信号量**

}CombineTask;

// model related param

float \*fea\_average\_;

float \*fea\_variance\_;

char\*\* param\_buffer\_;

char\*\* json\_buffer\_;

int vocab\_size\_;

int fea\_dim\_;

int\* device\_type\_; **// device**

int\* device\_id\_;

int max\_frame\_len\_;

int max\_batch\_size\_;

bool is\_half\_float\_;

mx\_uint64 param\_buffer\_size\_[4];

mlpflow::MXModel\* pMXModel\_;

mlpflow::MXSession\* pMXSession\_;

DataInfo \* pNDArrayHandle\_;

int inst\_id\_;

bool decFinish\_;

};

* 该类负责数据的push，然后完成数据的信号量以及输出缓存区的开辟设置
* 每个外部输入的任务需要进行管理，保留信号量和缓存区地址
* 外部调用wFeaDecFeatureReadLite，本质是一个挨着一个从队列里获取信号量，然后取对应的后验输出概率

外部送音频数据，在while循环中通过信号量等数据时，存在错过信号量的可能性，导致解码线程卡死（mx\_executor中pop数据时会出现）

如何实现队列里面的信号量和内部pop数据同步

信号加一个mask标记，如果已经pop则将信号的mask标记为true，则当前外部对该信号量进行wait时，就不需要等待了，而且此时该信号量绑定的buffer，后验概率的数据肯定已经传递到buffer中，可以直接利用该数据进行dec解码

1. EDGen调用的流程关系

* w\_rec模块

AUDIO\_END？

w\_rec(wAudioWriteImpLite)

w\_fea(wFeaAudioWriteLite)

w\_lite\_mlp(pushTask)

w\_fea(wFeaLiteWaitComplete)

Yes

* w\_lite\_mlp模块

MXHybridCNNPredictor::initialize

MXHybridCNNPredictor::resAdd

MXHybridCNNPredictor::creatInst

MXHybridCNNPredictor::resDelete

MXHybridCNNPredictor::uninitialize

MXHybridCNNPredictor::destroyInst

图2 w\_lite\_mlp进程级调用流程图

MXHybridCNNPredictor::start

MXHybridCNNPredictor::pushTask

MXHybridCNNPredictor::flushTask

MXHybridCNNPredictor::reset

MXHybridCNNPredictor::stop

MXHybridCNNPredictor::waitComplete

**循环push数据**

ThreadTask

MXHybridCNNPredictor::beamSearch

图3 w\_lite\_mlp回话级调用流程图