Generate trt engine from onnx file using c++ APIs

# 1. Build TensorRT engine

nvinfer1::createInferBuilder

builder->createNetworkV2(explicitBatch)

builder->createBuilderConfig()

nvonnxparser::createParser

constructNetwork(parser, builder, network, config);

builder->buildEngineWithConfig(\*network, \*config)

mEngine->serialize()

# 2. construct network using TensorRT APIs

Like other AI frameworks, TensorRT provide APIs for user to construct network for inference

## 2.1 save weights from pytorch

电脑屏幕截图

描述已自动生成

文本

描述已自动生成

## 2.2 load weights from 2.1



## define the network using APIs

1.Refer the original network and find the right APIs in TenorRT.

文本

描述已自动生成

2.Do not forget non weights layers such as activation, pooling, slice, etc.

3.Make sure input and output tensor of your added layers correct.

4.If you add the customized plugin, followed this:

文本

描述已自动生成

5.unmark and mark the graph outputs.

文本

描述已自动生成

# 3. define customized plugin

## 3.1 implement plugin class

A custom layer is implemented by extending the class IPluginCreator and one of TensorRT’s base classes for plugins. [IPluginCreator](https://docs.nvidia.com/deeplearning/sdk/tensorrt-api/c_api/classnvinfer1_1_1_i_plugin_creator.html) is a creator class for custom layers using which users can get plugin name, version, and plugin field parameters.

You should inherent a class from following and override the virtual methods.

[IPluginV2Ext](https://docs.nvidia.com/deeplearning/sdk/tensorrt-api/c_api/classnvinfer1_1_1_i_plugin_v2_ext.html)

[IPluginV2DynamicExt](https://docs.nvidia.com/deeplearning/sdk/tensorrt-api/c_api/classnvinfer1_1_1_i_plugin_v2_dynamic_ext.html)

// IPluginV2 methods

const char\* getPluginType() const override;

const char\* getPluginVersion() const override;

int getNbOutputs() const override;

int initialize() override;

void terminate() override;

size\_t getSerializationSize() const override;

void serialize(void\* buffer) const override;

void destroy() override;

void setPluginNamespace(const char\* libNamespace) override;

const char\* getPluginNamespace() const override;

电脑屏幕的截图

描述已自动生成

The four methods that are affected by dynamic shapes are:

* getOutputDimensions
* supportsFormatCombination
* configurePlugin
* enqueue

Most of the pure virtual methods are common to plugins. The main methods that affect INT8 I/O are:

* supportsFormatCombination
* configurePlugin
* enqueue

## 3.2 build shared library using cuda

put all the plugin execution in .cu file and compile it to .so file(libFasterRCNNKernels.so). refer it when you use the customized plugins.

Implementation of enqueue

文本

描述已自动生成

In FasterRCNNKernels.cu, roiAlign call roiAlign\_kernel

文本

描述已自动生成

# 4.verify inference result

## 4.1 image\_preprocess

1. Resize and pad

resize\_to\_bgrmat\_with\_ratio(cv::Mat& input\_image, int toH, int toW, bool keep\_ratio=true)

2. normalize

image\_normalize(cv::Mat& input\_image, const float\* img\_mean, const float\* img\_std)

3.permute: HWC🡪CHW

## 4.2 post\_process

Intra class nms:

class\_nms(std::vector<BBox>& bboxes, float iou\_threshold = 0.5, float score\_threshold = 0.05)

# 5. Int8 quantization

TensorRT built-in support fp16 quantization, you only need to set the IBuilderConfiger:

config->setFlag(BuilderFlag::kFP16);

w.r.t int8 quantization you need to do follows:

config->setFlag(BuilderFlag::kINT8);

samplesCommon::setAllTensorScales(network.get(), 127.0f, 127.0f);

If you want to do calibration, replace the above by this:

assert(builder->platformHasFastInt8());

config->setFlag(BuilderFlag::kINT8);

Int8EntropyCalibrator2\* calibrator = new Int8EntropyCalibrator2(mParams.batchSize, inputSize, inputSize, "/home/ubuntu/data/coco/val2017/",

"int8calib\_fasterrcnn.table", input->getName());

config->setInt8Calibrator(calibrator);

Rewrite two functions in class Int8EntropyCalibrator2:

getBatchSize() getBatch()

in getBatch() you should implement the image file loading and image preprocess, the image preprocess shall identical to which in model inference.

# 6.references

<https://github.com/dulvqingyunLT/trt_projects>

<https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html#add_custom_layer>