作业第七章

作业内容是使用 TensorRT Python API 手动搭建 BERT 模型。同学们不需要从零 开始搭建，课程组已经搭好了一个模型结构框架，同学们进行相应的填充即可。

模型架构框架：https://github.com/shenlan2017/TensorRT

一. 文件信息

1. model2onnx.py 使用 pytorch 运行 bert 模型，生成 demo 输入输出和 onnx 模型
2. onnx2trt.py　将 onnx，使用 onnx-parser 转成 trt 模型，并 infer
3. builder.py　输入 onnx 模型，并进行转换
4. trt_helper.py　对 trt 的 api 进行封装，方便调用
5. calibrator.py int8 calibrator 代码
6. 基础款 LayerNormPlugin.zip　用于学习的 layer_norm_plugin

二. 模型信息

2.1 介绍

1. 标准 BERT 模型，12 层, hidden_size = 768
2. 不考虑 tokensizer 部分，输入是 ids，输出是 score
3. 为了更好的理解，降低作业难度，将 mask 逻辑去除，只支持 batch=1 的输入

BERT 模型可以实现多种 NLP 任务，作业选用了 fill-mask 任务的模型

```
输入：
The capital of France, [mask], contains the Eiffel Tower.

topk10 输出：
The capital of France, paris, contains the Eiffel Tower.
The capital of France, lyon, contains the Eiffel Tower.
The capital of France,, contains the Eiffel Tower.
The capital of France, tolilleulouse, contains the Eiffel Tower.
The capital of France, marseille, contains the Eiffel Tower.
The capital of France, orleans, contains the Eiffel Tower.
The capital of France, strasbourg, contains the Eiffel Tower.
The capital of France, nice, contains the Eiffel Tower.
The capital of France, cannes, contains the Eiffel Tower.
The capital of France, versailles, contains the Eiffel Tower.
```

2.2 输入输出信息

输入

1. input_ids[1,-1]：int 类型，input ids，从 BertTokenizer 获得
2. token_type_ids[1, -1]：int 类型，全 0
3. position_ids[1, -1]：int 类型，[0, 1, ..., len(input_ids) - 1]

输出

1. logit[1, -1, 768]

三. 作业内容

3.1 进行 fp16 加速并测试速度

及格标准：设置 build_config，对模型进行 fp16 优化；
优秀标准：编写 fp16 版本的 layer_norm 算子，使模型最后运行 fp16 版本的 layer_norm 算子。

3.2 进行 int8 加速并测试速度

1. 完善 calibrator.py 内的 todo 函数，使用 calibrator_data.txt 校准集，对模型进行 int8 量化加速。

四. 深度思考

4.1 还有那些算子能合并?

1. emb_layernorm 模块，3 个 embedding 和一个 layer_norm，是否可以合并到一个 kernel 中?
2. self_attention_layer 中，softmax 和 scale 操作，是否可以合并到一个 kernel 中?
3. self_attention_layer，要对 qkv 进行三次矩阵乘和 3 次转置。三个矩阵乘是否可以合并到一起，相应三个转置是否可以? 如果合并了，那么后面的 q*k* 和 *attn*v，该怎么计算?
4. self_output_layer 中，add和layer_norm层是否可以合并?

以上问题的答案，见 https://github.com/NVIDIA/TensorRT/tree/release/6.0/demo/BERT

4.2 除了上面那些，还能做那些优化?

1. 增加 mask 逻辑，多 batch 一起 inference，padding 的部分会冗余计算。比如一句话 10 个字，第二句 100 个字，那么 padding 后的大小是[2, 100]，会有 90 个字的冗余计算。这部分该怎么优化? 除了模型内部优化，是否还可以在外部拼帧逻辑进行优化?
2. self_attention_layer 层，合并到一起后的 QkvToContext 算子，是否可以支持 fp16 计算? int8 呢?

以上问题答案，见 https://github.com/NVIDIA/TensorRT/tree/release/8.2/demo/BERT