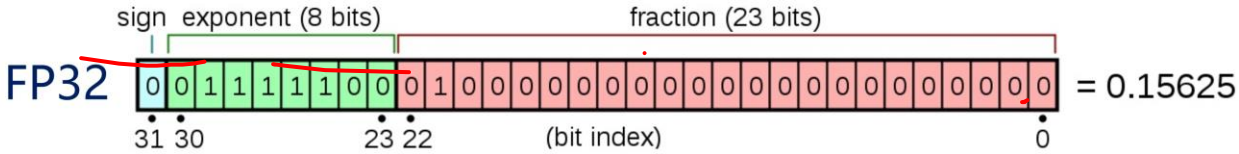
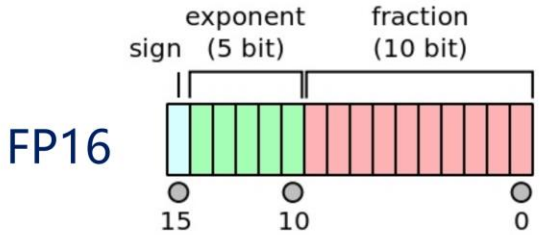
TensorRT FP16加速

深蓝学院课程视频《TensorRT INT8量化加速》章节的查漏补缺

TensorRT FP16加速-FP16是什么



$$\text{Value} = sign \times 2^{(exponent-127)} \times (1 + fraction)$$



$$\text{Value} = sign \times 2^{(exponent-15)} \times (1 + fraction)$$

TensorRT FP16加速

config->setFlag(BuilderFlag::kFP16);

builder->platformHasFastFp16()　　builder->platformHasFastInt8()

4. Hardware And Precision

The following table lists NVIDIA hardware and which precision modes each hardware supports. TensorRT supports all NVIDIA hardware with capability SM 5.0 or higher. It also lists the availability of Deep Learning Accelerator (DLA) on this hardware. Refer to the following tables for the specifics.

Note: Support for CUDA Compute Capability version 3.0 has been removed. Support for CUDA Compute Capability versions below 5.0 may be removed in a future release and is now deprecated.

Table 4. Supported hardware

CUDA Compute Capability	Example Device	TF32	FP32	FP16	INT8	FP16 Tensor Cores	INT8 Tensor Cores	DLA
8.6	NVIDIA A10	Yes	Yes	Yes	Yes	Yes	Yes	No
8.0	NVIDIA A100/GA100 GPU	Yes	Yes	Yes	Yes	Yes	Yes	No
7.5	Tesla T4	No	Yes	Yes	Yes	Yes	Yes	No
7.2	Jetson AGX Xavier	No	Yes	Yes	Yes	Yes	Yes	Yes
7.0	Tesla V100	No	Yes	Yes	Yes	Yes	No	No
6.2	Jetson TX2	No	Yes	Yes	No	No	No	No
6.1	Tesla P4	No	Yes	No	Yes	No	No	No
6.0	Tesla P100	No	Yes	Yes	No	No	No	No
5.3	Jetson TX1	No	Yes	Yes	No	No	No	No
5.2	Tesla M4	No	Yes	No	No	No	No	No
5.0	Quadro K2200	No	Yes	No	No	No	No	No

TensorRT Plugin FP16加速-大纲

1. 编写Plugin注意事项

   1.1 Enqueue 函数增加half版本

   1.2 supportsFormatCombination函数

2. fp16模型，输入设置为float类型还是half类型？

3. 模型配合混合精度训练，否则可能会出现溢出问题

https://github.com/NVIDIA/TensorRT/blob/7.2.1/plugin/skipLayerNormPlugin/skipLayerNormPlugin.cpp

Enqueue 函数增加half版本

```
657     if (iType == DataType::kFLOAT)
658     {
659         const auto input = static_cast<const float*>(inputs[0]);
660         const auto skip = static_cast<const float*>(inputs[1]);
661         auto output = static_cast<float*>(outputs[0]);
662         const auto bias = static_cast<const float*>(mBiasDev.get());
663         const auto beta = static_cast<const float*>(mBetaDev.get());
664         const auto gamma = static_cast<const float*>(mGammaDev.get());
665         if (mHasBias)
666         {
667             status
668                 = computeSkipLayerNorm<float, true>(stream, static_cast<int>(mLd), inputVolume, input, skip, beta, gamma, output, bias);
669         }
670         else
671         {
672             status
673                 = computeSkipLayerNorm<float, false>(stream, static_cast<int>(mLd), inputVolume, input, skip, beta, gamma, output, bias);
674         }
675     }
676     else if (iType == DataType::kHALF)
677     {
678         const auto input = static_cast<const half*>(inputs[0]);
679         const auto skip = static_cast<const half*>(inputs[1]);
680         auto output = static_cast<half*>(outputs[0]);
681         const auto bias = static_cast<const half*>(mBiasDev.get());
682         const auto beta = static_cast<const half*>(mBetaDev.get());
683         const auto gamma = static_cast<const half*>(mGammaDev.get());
684         if (mHasBias)
685         {
686             status = computeSkipLayerNorm<half, true>(stream, static_cast<int>(mLd), inputVolume, input, skip, beta, gamma, output, bias);
687         }
688         else
689         {
690             status
691                 = computeSkipLayerNorm<half, false>(stream, static_cast<int>(mLd), inputVolume, input, skip, beta, gamma, output, bias);
692         }
693     }
694     else if (iType == DataType::kINT8)
```

supportsFormatCombination函数

Int8模式暂不考虑

1. 保证输入输出类型一致
2. 要求输入输出类型与mType一致

```
133 bool SkipLayerNormPluginDynamic::supportsFormatCombination(
134     int pos, const PluginTensorDesc* inOut, int nbInputs, int nbOutputs)
135 {
136     assert(nbInputs == 2);
137     assert(nbOutputs == 1);
138 
139     const PluginTensorDesc& in = inOut[pos];
140     if (pos == 0)
141     {
142         // Since H = W = 1, we can report CHWx for any x
143         if (mType == DataType::kINT8)
144         {
145             // won't work for hiddensize too small!
146             TensorFormat myFmt = TensorFormat::kCHW32;
147             if (mLd < 32)
148             {
149                 myFmt = TensorFormat::kCHW4;
150                 gLogVerbose << "SkipLayerNormDQQ: TensorFormat CHW4"
151                             << " for LD=" << mLd << std::endl;
152             }
153             else
154             {
155                 gLogVerbose << "SkipLayerNormDQQ: TensorFormat CHW32"
156                             << " for LD=" << mLd << std::endl;
157             }
158             // TODO do we need to check if the vectorization divides mLd?
159             return ((in.type == mType) && (in.format == myFmt));
160         }
161         return (in.type == mType) && (in.format == TensorFormat::kLINEAR);
162     }
163     const PluginTensorDesc& prev = inOut[pos - 1];
164 
165     return in.type == prev.type && in.format == prev.format;
166 }
167 
```

fp16模型，输入设置为float类型还是half类型？

建议输入设置成float

一段demo代码，input=>relu=>plugin=>relu=>output

```
33
34    // the input type is float.
35    // if enable fp16, input will be casted to fp16 firstly
36    auto input = network->addInput("input", DataType::kFLOAT, Dims3{-1, -1, dim});
37
38    nvinfer1::Weights w_gamma = nvinfer1::Weights{DataType::kFLOAT, gamma.data_ptr<float>(), (int)gamma.numel()};
39    nvinfer1::Weights w_beta = nvinfer1::Weights{DataType::kFLOAT, beta.data_ptr<float>(), (int)beta.numel()};
40
41    // add relu to cast input to fp16.
42    auto relu1 = network->addActivation(*input, ActivationType::kRELU);
43
44    auto outputs = add_layer_norm_dynamic_plugin(network.get(), {relu1->getOutput(0)}, data_type, dim, w_gamma, w_beta);
45
46    auto relu2 = network->addActivation(*outputs[0], ActivationType::kRELU);
47
48    network->markOutput(*relu2->getOutput(0));
49
```

fp32

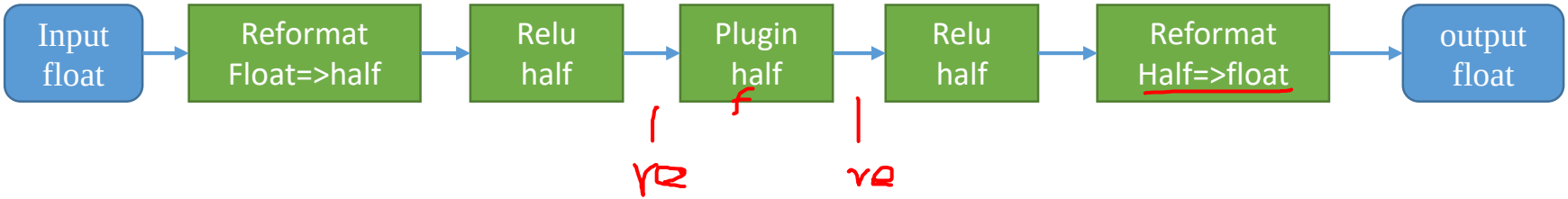
fp16

fp16模型，输入设置为float类型还是half类型？

建议输入设置成float

Build结束后的log如下

```
[03/11/2022-11:14:03] [V] [TRT] Engine Layer Information:
[03/11/2022-11:14:03] [V] [TRT] Layer(Reformat): (Unnamed Layer* 0) [Activation] input reformatter 0, Tactic: 1002, input[Float(-2147483644,-2147483643,64)] -> (Unnamed Layer* 0) [Activation] reformatted input 0[Half(-2147483644,-2147483643,64)]
[03/11/2022-11:14:03] [V] [TRT] Layer(Activation): (Unnamed Layer* 0) [Activation], Tactic: 0, (Unnamed Layer* 0) [Activation] reformatted input 0[Half(-2147483644,-2147483643,64)] -> (Unnamed Layer* 0) [Activation]_output[Half(-2147483644,-2147483643,64)]
[03/11/2022-11:14:03] [V] [TRT] Layer(PluginV2): 2 [SiluPlugin], Tactic: 0, (Unnamed Layer* 0) [Activation]_output[Half(-2147483644,-2147483643,64)] -> (Unnamed Layer* 1) [PluginV2DynamicExt]_output_0[Half(-2147483644,-2147483643,64)]
[03/11/2022-11:14:03] [V] [TRT] Layer(Activation): (Unnamed Layer* 2) [Activation], Tactic: 0, (Unnamed Layer* 1) [PluginV2DynamicExt]_output_0[Half(-2147483644,-2147483643,64)] -> (Unnamed Layer* 2) [Activation] output to be reformatted 0[Half(-2147483644,-2147483643,64)]
[03/11/2022-11:14:03] [V] [TRT] Layer(Reformat): (Unnamed Layer* 2) [Activation] output reformatter 0, Tactic: 1002, (Unnamed Layer* 2) [Activation] output to be reformatted 0[Half(-2147483644,-2147483643,64)] -> (Unnamed Layer* 2) [Activation]_output[Float(-2147483644,-2147483643,64)]
[03/11/2022-11:14:03] [V] [TRT] Deserialize required 3099 microseconds.
```



模型配合混合精度训练，否则可能会出现溢出问题