关于模型转换

目录

背景

为了实现跨平台的等价推理。

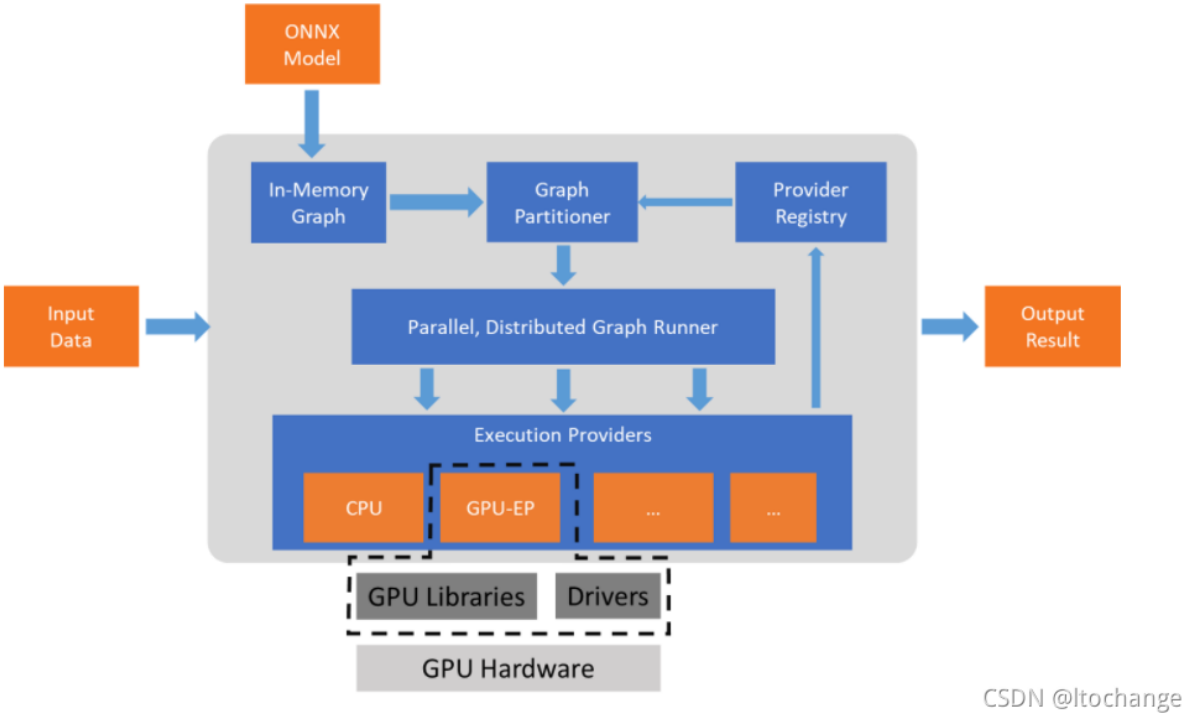
已知：

- pth为使用pytorch框架训练得到的模型参数文件，可在x86平台上推理
- onnx可以跨平台实现推理，最后用trt引擎加速
- tidl为TI公司为TDA4VM处理器专门设计的一套针对神经网络的嵌入式SDK

关于pth-onnx-tensorRT

onnx（Open Neural Network Exchange）：微软和Facebook开发，它可以成为不同模型（pytorch、TensorFlow）之间的桥梁。

onnx runtime是将onnx模型部署到生产环境的跨平台高性能运行引擎；



from csdn

tensorrt是nvidia开发的神经网络前向推理（eval状态）的C++库，只需把模型提供给该框架即可实现加速，加速方式包括：

- 权重量化：例如模型量化为INT8
- 层与张量融合、动态调整内核、张量现存

比较：

gpu测试：

使用gpu		pytorch	onnxruntime	tensorrt
	单条时间	14.2ms	1.5ms	2.5ms
batch=1	显存占用	1290M	1420M	856M
	gpu利用率	14%	68%	62%
	单条时间	1.8ms	0.2ms	0.3ms
batch=8	显存占用	1356M	1426	856M
	gpu利用率	18%	85%	83%
	单条时间	0.9ms	0.13ms	0.17ms
batch=16	显存占用	1446M	1434	858M
	gpu利用率	25%	88%	94%

1. onnxruntime与tensortrt的gpu利用率要比pytorch高很多
2. tensorrt在未作量化的情况下，显存占用更小
3. 随着batch的增大，速度提升越来越不明显

Onnx的部署、推理过程

pth-onnx格式转化 Python | 收起

```python
import torch.onnx
# 转换的onnx格式的名称，文件后缀需为.onnx
onnx_file_name = "xxxxxx.onnx"
# 我们需要转换的模型，将torch_model设置为自己的模型
model = torch_model
# 加载权重，将model.pth转换为自己的模型权重
# 如果模型的权重是使用多卡训练出来，我们需要去除权重中多的module. 具体操作可以见5.4节
model = model.load_state_dict(torch.load("model.pth"))
# 导出模型前，必须调用model.eval()或者model.train(False)
model.eval()
# dummy_input就是一个输入的实例，仅提供输入shape、type等信息
batch_size = 1 # 随机的取值，当设置dynamic_axes后影响不大
dummy_input = torch.randn(batch_size, 1, 224, 224, requires_grad=True)
# 这组输入对应的模型输出
output = model(dummy_input)
# 导出模型
torch.onnx.export(model,         # 模型的名称
                  dummy_input,   # 一组实例化输入
                  onnx_file_name,    # 文件保存路径/名称
                  export_params=True,         #  如果指定为True或默认，参数也会被导出. 如果你要导出一个没训练过的就设为 False.
                  opset_version=10,           # ONNX 算子集的版本，当前已更新到15
                  do_constant_folding=True,  # 是否执行常量折叠优化
                  input_names = ['input'],    # 输入模型的张量的名称
                  output_names = ['output'], # 输出模型的张量的名称
                  # dynamic_axes将batch_size的维度指定为动态，
                  # 后续进行推理的数据可以与导出的dummy_input的batch_size不同
                  dynamic_axes={'input' : {0 : 'batch_size'},
                                'output' : {0 : 'batch_size'}})
```

from https://datawhalechina.github.io/thorough-pytorch/%E7%AC%AC%E4%B9%9D%E7%AB%A0/9.1%20%E4%BD%BF%E7%94%A8ONNX%E8%BF%9B%E8%A1%8C%E9%83%A8%E7%BD%B2%E5%B9%B6%E6%8E%A8%E7%90%86.html#id3

转换为onnx后，可通过Netron可视化。

使用onnx runtime进行推理：

Python | 收起

```python
# 导入onnxruntime
import onnxruntime
# 需要进行推理的onnx模型文件名称
onnx_file_name = "xxxxxx.onnx"

# onnxruntime.InferenceSession用于获取一个 ONNX Runtime 推理器
ort_session = onnxruntime.InferenceSession(onnx_file_name)

# 构建字典的输入数据，字典的key需要与我们构建onnx模型时的input_names相同
# 输入的input_img 也需要改变为ndarray格式
ort_inputs = {'input': input_img}
# 我们更建议使用下面这种方法,因为避免了手动输入key
# ort_inputs = {ort_session.get_inputs()[0].name:input_img}

# run是进行模型的推理，第一个参数为输出张量名的列表，一般情况可以设置为None
# 第二个参数为构建的输入值的字典
# 由于返回的结果被列表嵌套，因此我们需要进行[0]的索引
ort_output = ort_session.run(None,ort_inputs)[0]
# output = {ort_session.get_outputs()[0].name}
# ort_output = ort_session.run([output], ort_inputs)[0]
```

onnx变量属性（att）详解：

model	表示整个onnx模型，包含图结构和解析器格式
model.graph	表示图结构，通常是netron看到的结构
model.graph.node	表示节点，包括graph中的conv、bn
model.graph.initializer	权重类数据
model.graph.input	整个模型的输入储存在这（array）
model.graph.output	整个模型的输出（array）