Shape 层

- 初始示例代码（TensorRT8 中的用法）
- TensorRT7 + static shape 模式中的用法
- TensorRT7 + dynamic shape 模式中的用法
- TensorRT6 中的用法（已废弃）[TODO]

初始示例代码（TensorRT8 中的用法）

```
import numpy as np
from cuda import cudart
import tensorrt as trt

nIn, cIn, hIn, wIn = 1, 3, 4, 5  # 输入张量 NCHW
data = np.arange(cIn, dtype=np.float32).reshape(cIn, 1, 1) * 100 + np.arange(hIn).reshape(1, hIn, 1) *
10 + np.arange(wIn).reshape(1, 1, wIn)  # 输入数据
data = data.reshape(nIn, cIn, hIn, wIn).astype(np.float32)

np.set_printoptions(precision=8, linewidth=200, suppress=True)
cudart.cudaDeviceSynchronize()

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
config = builder.create_builder_config()
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (nIn, cIn, hIn, wIn))
#------------------------------------------------------------ --------------------# 替换部分
shapeLayer = network.add_shape(inputT0)
#------------------------------------------------------------ --------------------# 替换部分
network.mark_output(shapeLayer.get_output(0))
engineString = builder.build_serialized_network(network, config)
engine = trt.Runtime(logger).deserialize_cuda_engine(engineString)
context = engine.create_execution_context()
_, stream = cudart.cudaStreamCreate()

inputH0 = np.ascontiguousarray(data.reshape(-1))
outputH0 = np.empty(context.get_binding_shape(1), dtype=trt.nptype(engine.get_binding_dtype(1)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)

cudart.cudaMemcpyAsync(inputD0, inputH0.ctypes.data, inputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
context.execute_async_v2([int(inputD0), int(outputD0)], stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
cudart.cudaStreamSynchronize(stream)

print("inputH0 :", data.shape)
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)

cudart.cudaStreamDestroy(stream)
cudart.cudaFree(inputD0)
```

```
cudart.cudaFree(outputD0)
```

- 输入张量形状 (1,3,4,5)

$$\left[\left[\left[\begin{matrix}1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1.\end{matrix}\right]\left[\begin{matrix}1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1.\end{matrix}\right]\left[\begin{matrix}1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1. \\ 1. & 1. & 1. & 1. & 1.\end{matrix}\right]\right]\right]$$

- 输出张量形状 (4,)

$$\left[\begin{matrix}1 & 3 & 4 & 5\end{matrix}\right]$$

- 必须使用 explicit batch 模式，否则报错

```
[TRT] [E] 3: [network.cpp::addShape::1163] Error Code 3: API Usage Error (Parameter check failed at:
optimizer/api/network.cpp::addShape::1163, condition: !hasImplicitBatchDimension()
)
```

TensorRT7 + static shape 模式中的用法

```
import numpy as np
from cuda import cudart
import tensorrt as trt

nIn, cIn, hIn, wIn = 1, 3, 4, 5
data = np.arange(cIn, dtype=np.float32).reshape(cIn, 1, 1) * 100 + np.arange(hIn).reshape(1, hIn, 1) *
10 + np.arange(wIn).reshape(1, 1, wIn)
data = data.reshape(nIn, cIn, hIn, wIn).astype(np.float32)

np.set_printoptions(precision=8, linewidth=200, suppress=True)
cudart.cudaDeviceSynchronize()

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
config = builder.create_builder_config()
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (nIn, cIn, hIn, wIn))
shapeLayer = network.add_shape(inputT0)
network.mark_output(shapeLayer.get_output(0))
engine = builder.build_engine(network, config)  # 使用旧版的 engine 生成 API
context = engine.create_execution_context()
_, stream = cudart.cudaStreamCreate()

outputH0 = np.empty(context.get_binding_shape(0), dtype=trt.nptype(engine.get_binding_dtype(0)))  # 不需
要绑定输入张量（engine中已经包含了输入张量的形状信息）
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)

context.execute_async_v2([int(outputD0)], stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
cudart.cudaStreamSynchronize(stream)

print("inputH0 :", data.shape)
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)
```

```
cudart.cudaStreamDestroy(stream)
cudart.cudaFree(outputD0)
```

- 输出张量形状 (4,)，结果与初始示例代码相同
- 该方法在 TensorRT8 中也适用，但是必须传入输入张量

TensorRT7 + dynamic shape 模式中的用法

```
import numpy as np
from cuda import cudart
import tensorrt as trt

nIn, cIn, hIn, wIn = 1, 3, 4, 5
data = np.arange(cIn, dtype=np.float32).reshape(cIn, 1, 1) * 100 + np.arange(hIn).reshape(1, hIn, 1) *
10 + np.arange(wIn).reshape(1, 1, wIn)
data = data.reshape(nIn, cIn, hIn, wIn).astype(np.float32)

np.set_printoptions(precision=8, linewidth=200, suppress=True)
cudart.cudaDeviceSynchronize()

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
profile = builder.create_optimization_profile()
config = builder.create_builder_config()
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (-1, -1, -1, -1))
profile.set_shape(inputT0.name, (1, 1, 1, 1), (nIn, cIn, hIn, wIn), (nIn * 2, cIn * 2, hIn * 2, wIn *
2))
config.add_optimization_profile(profile)
shapeLayer = network.add_shape(inputT0)
network.mark_output(shapeLayer.get_output(0))
engine = builder.build_engine(network, config)
context = engine.create_execution_context()
context.set_binding_shape(0, data.shape)  # 需要绑定真实输入张量形状
_, stream = cudart.cudaStreamCreate()

inputH0 = np.ascontiguousarray(data.reshape(-1))  # 需要传入真实输入张量
outputH0 = np.empty(context.get_binding_shape(1), dtype=trt.nptype(engine.get_binding_dtype(1)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)

cudart.cudaMemcpyAsync(inputD0, inputH0.ctypes.data, inputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
context.execute_async_v2([int(inputD0), int(outputD0)], stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
cudart.cudaStreamSynchronize(stream)

print("inputH0 :", data.shape)
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)

cudart.cudaStreamDestroy(stream)
cudart.cudaFree(inputD0)
cudart.cudaFree(outputD0)
```

- 输出张量形状 (4,)，结果与初始示例代码相同
- 该方法在 TensorRT 中也适用

TensorRT6 中的用法

```
import numpy as np
from cuda import cudart
import tensorrt as trt

nIn, cIn, hIn, wIn = 1, 3, 4, 5
data = np.arange(cIn, dtype=np.float32).reshape(cIn, 1, 1) * 100 + np.arange(hIn).reshape(1, hIn, 1) *
10 + np.arange(wIn).reshape(1, 1, wIn)
data = data.reshape(nIn, cIn, hIn, wIn).astype(np.float32)

np.set_printoptions(precision=8, linewidth=200, suppress=True)
cudart.cudaDeviceSynchronize()

logger = trt.Logger(trt.Logger.ERROR)
builder = trt.Builder(logger)
network = builder.create_network(1 << int(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH))
config = builder.create_builder_config()
config.max_workspace_size = 1 << 30
constantLayer = network.add_constant(data.shape, data)

shapeLayer = network.add_shape(constantLayer.get_output(0))
network.mark_output(shapeLayer.get_output(0))
network.mark_output_for_shapes(shapeLayer.get_output(0))  # “标记形状张量作为输出”的专用方法
engine = builder.build_engine(network, config)  # 使用旧版 API
context = engine.create_execution_context()
_, stream = cudart.cudaStreamCreate()

print(engine.num_bindings)
print(engine.binding_is_input(0))
print(context.all_binding_shapes_specified, context.all_shape_inputs_specified)
#print(context.get_binding_shape(0))
print(engine.get_binding_shape(0))
print(engine.get_binding_dtype(0))

#print(context.get_shape(0))

outputH0 = np.empty(engine.get_binding_shape(0), dtype=trt.nptype(engine.get_binding_dtype(0)))  # 不需要
绑定输入张量（engine中已经包含了输入张量的形状信息）
print("here!")
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)
print("here!")
context.execute_async(1, [], stream)
print("here!")
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
print("here!")
cudart.cudaStreamSynchronize(stream)
print("here!")

print("inputH0 :", data.shape)
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)
```

```
cudart.cudaStreamDestroy(stream)
cudart.cudaFree(outputD0)
```

- 注意，TensorRT 7 中两种方法均可使用，TensorRT 6 中只能使用专用方法，若在 TensorRT 6 中使用常规方法，则会得到如下报错：

> [TensorRT] ERROR: (Unnamed Layer* 1) [Shape]: ShapeLayer output tensor ((Unnamed Layer* 1) [Shape]_output) used as an execution tensor, but must be used only as shape tensor.
> build engine failed.