Slice 层

初始示例代码

```
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.float32, (nIn, cIn, hIn, wIn))
#---------------------------------------------------------- --------------------# 替换部分
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (1, 2, 3, 4), (1, 1, 1, 1))
#---------------------------------------------------------- --------------------# 替换部分
network.mark_output(sliceLayer.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)，百位、十位、个位分别表示 CHW 维编号

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 11. & 12. & 13. & 14. \\ 20. & 21. & 22. & 23. & 24. \\ 30. & 31. & 32. & 33. & 34. \end{bmatrix} \begin{bmatrix} 100. & 101. & 102. & 103. & 104. \\ 110. & 111. & 112. & 113. & 114. \\ 120. & 121. & 122. & 123. & 124. \\ 130. & 131. & 132. & 133. & 134. \end{bmatrix} \begin{bmatrix} 200. & 201. & 202. & 203. & 204. \\ 210. & 211. & 212. & 213. & 214. \\ 220. & 221. & 222. & 223. & 224. \\ 230. & 231. & 232. & 233. & 234. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 输出张量形状 (1,2,3,4)，以 (0,0,0) 元素为起点，切出 (1,2,3,4) 形状的张量，各维上的步长为 (1,1,1)

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 0. & 1. & 2. & 3. \\ 10. & 11. & 12. & 13. \\ 20. & 21. & 22. & 23. \end{bmatrix} \begin{bmatrix} 100. & 101. & 102. & 103. \\ 110. & 111. & 112. & 113. \\ 120. & 121. & 122. & 123. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

---

start & shape & stride

```
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (0, 0, 0, 0), (0, 0, 0, 0))
sliceLayer.start = (0, 0, 0, 0)  # 重设起点位置
sliceLayer.shape = (1, 2, 3, 4)  # 重设输出张量形状
sliceLayer.stride = (1, 1, 1, 1)  # 重设切割步长，1 为连续提取元素
```

- 输出张量形状 (1,2,3,4)，结果与初始示例代码相同

---

mode

```
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (1, 2, 3, 4), (1, 2, 2, 2))
sliceLayer.mode = trt.SliceMode.WRAP
```

- 指定 mode=trt.SliceMode.WRAP，输出张量形状 (1,2,3,4)，超出边界的元素从起始元素继续切片

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 0. & 2. & 4. & 1. \\ 20. & 22. & 24. & 21. \\ 0. & 2. & 4. & 1. \end{bmatrix} \begin{bmatrix} 200. & 202. & 204. & 201. \\ 220. & 222. & 224. & 221. \\ 200. & 202. & 204. & 201. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 指定 mode=trt.SliceMode.CLAMP，输出张量形状 (1,2,3,4)，超出边界的元素取边界值

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 0. & 2. & 4. & 4. \\ 20. & 22. & 24. & 24. \\ 30. & 32. & 34. & 34. \end{bmatrix} \begin{bmatrix} 200. & 202. & 204. & 204. \\ 220. & 222. & 224. & 224. \\ 230. & 232. & 234. & 234. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 指定 mode=trt.SliceMode.FILL，输出张量形状 (1,2,3,4)，超出边界的元素取固定值，默认值 0
- 需要设置其他固定值的情况，参见后面的 fill mode + set_input 部分

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 0. & 2. & 4. & 0. \\ 20. & 22. & 24. & 0. \\ 0. & 0. & 0. & 0. \end{bmatrix} \begin{bmatrix} 200. & 202. & 204. & 0. \\ 220. & 222. & 224. & 0. \\ 0. & 0. & 0. & 0. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- 指定 mode=trt.SliceMode.REFLECT，输出张量形状 (1,2,3,4)，超出边界的元素折返取值

$$\left[\left[\begin{bmatrix} 0. & 2. & 4. & 2. \\ 20. & 22. & 24. & 22. \\ 20. & 22. & 24. & 22. \end{bmatrix} \begin{bmatrix} 200. & 202. & 204. & 0. \\ 220. & 222. & 224. & 0. \\ 220. & 222. & 224. & 0. \end{bmatrix}\right]\right]$$

- 可用的模式

trt.SliceMode 名	说明
DEFAULT	默认模式，超出边界就报错
WRAP	data[i] = data[i%w]
CLAMP	data[i] = data[w-1]
FILL	data[i] = fillValue
REFLECT	data[i] = data[(w-1-i%w)*(i/w%2)+(i%w)*(1-i/w%2)]

---

set_input

fill mode + set_input

```python
constantLayer = network.add_constant([1], np.array([-1], dtype=np.float32))
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (1, 2, 3, 4), (1, 2, 2, 2))
sliceLayer.mode = trt.SliceMode.FILL
sliceLayer.set_input(4, constantLayer.get_output(0))  # 在 4 号位上设置固定值
```

- 输出张量形状 (1,2,3,4)，超出边界的元素取固定值

$$\left[\left[\begin{bmatrix} 0. & 2. & 4. & -1. \\ 20. & 22. & 24. & -1. \\ -1. & -1. & -1. & -1. \end{bmatrix} \begin{bmatrix} 200. & 202. & 204. & -1. \\ 220. & 222. & 224. & -1. \\ -1. & -1. & -1. & -1. \end{bmatrix}\right]\right]$$

静态 set_input

```python
constantLayer0 = network.add_constant([4], np.array([0, 0, 0, 0], dtype=np.int32))
constantLayer1 = network.add_constant([4], np.array([1, 2, 3, 4], dtype=np.int32))
constantLayer2 = network.add_constant([4], np.array([1, 1, 1, 1], dtype=np.int32))
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (0, 0, 0, 0), (0, 0, 0, 0))
#sliceLayer.set_input(0,inputT0)                                                    # 0 号输入是被 slice 的张量
sliceLayer.set_input(1, constantLayer0.get_output(0))  # 2、3、4 号输入分别对应起点、形状和步长
sliceLayer.set_input(2, constantLayer1.get_output(0))
sliceLayer.set_input(3, constantLayer2.get_output(0))
```

- 输出张量形状 (1,2,3,4)，结果与初始示例代码相同

动态 set_input

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

nIn, cIn, hIn, wIn = 1, 3, 4, 5
data0 = 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)
data0 = data0.reshape(nIn, cIn, hIn, wIn).astype(np.float32)
```

```
data1 = np.array([0, 0, 0, 0], dtype=np.int32)
data2 = np.array([1, 2, 3, 4], dtype=np.int32)
data3 = np.array([1, 1, 1, 1], dtype=np.int32)

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()  # 需要使用 profile
config = builder.create_builder_config()
inputT0 = network.add_input('inputT0', trt.float32, (nIn, cIn, hIn, wIn))
inputT1 = network.add_input('inputT1', trt.int32, (4, ))
inputT2 = network.add_input('inputT2', trt.int32, (4, ))
inputT3 = network.add_input('inputT3', trt.int32, (4, ))
profile.set_shape_input(inputT1.name, (0, 0, 0, 0), (0, 1, 1, 1), (0, 2, 2, 2))  # 这里设置的不是 shape
input 的形状而是值
profile.set_shape_input(inputT2.name, (1, 1, 1, 1), (1, 2, 3, 4), (1, 3, 4, 5))
profile.set_shape_input(inputT3.name, (1, 1, 1, 1), (1, 1, 1, 1), (1, 1, 1, 1))
config.add_optimization_profile(profile)
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (0, 0, 0, 0), (0, 0, 0, 0))
#sliceLayer.set_input(0,inputT0)
sliceLayer.set_input(1, inputT1)
sliceLayer.set_input(2, inputT2)
sliceLayer.set_input(3, inputT3)
network.mark_output(sliceLayer.get_output(0))
engineString = builder.build_serialized_network(network, config)
engine = trt.Runtime(logger).deserialize_cuda_engine(engineString)
context = engine.create_execution_context()
context.set_shape_input(1, data1)  # 运行时绑定真实形状张量值
context.set_shape_input(2, data2)
context.set_shape_input(3, data3)
_, stream = cudart.cudaStreamCreate()

inputH0 = np.ascontiguousarray(data0.reshape(-1))
inputH1 = np.ascontiguousarray(np.zeros([4], dtype=np.int32).reshape(-1))  # 传形状张量数据可用垃圾值
inputH2 = np.ascontiguousarray(np.zeros([4], dtype=np.int32).reshape(-1))
inputH3 = np.ascontiguousarray(np.zeros([4], dtype=np.int32).reshape(-1))
outputH0 = np.empty(context.get_binding_shape(4), dtype=trt.nptype(engine.get_binding_dtype(4)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, inputD1 = cudart.cudaMallocAsync(inputH1.nbytes, stream)
_, inputD2 = cudart.cudaMallocAsync(inputH2.nbytes, stream)
_, inputD3 = cudart.cudaMallocAsync(inputH3.nbytes, stream)
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)

cudart.cudaMemcpyAsync(inputD0, inputH0.ctypes.data, inputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
cudart.cudaMemcpyAsync(inputD1, inputH1.ctypes.data, inputH1.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
cudart.cudaMemcpyAsync(inputD2, inputH2.ctypes.data, inputH2.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
cudart.cudaMemcpyAsync(inputD3, inputH3.ctypes.data, inputH3.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
context.execute_async_v2([int(inputD0), int(inputD1), int(inputD2), int(inputD3), int(outputD0)],
stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)
cudart.cudaStreamSynchronize(stream)
```

```
print("inputH0 :", data0.shape)
print(data0)
print("inputH1 :", data1.shape)
print(data1)
print("inputH2 :", data2.shape)
print(data2)
print("inputH3 :", data2.shape)
print(data3)
print("outputH0:", outputH0.shape)
print(outputH0)

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

- 输出张量形状 (1,2,3,4)，结果与静态 set_input 示例代码相同
- 建立网络时需要 profile，并在运行时绑定真实形状张量的值，否则会有下面几种报错：

```
# 没有用 profile
[TRT] [E] 4: [network.cpp::validate::2919] Error Code 4: Internal Error (Network has dynamic or shape
inputs, but no optimization profile has been defined.)
# 没有正确设置 profile
[TRT] [E] 4: [network.cpp::validate::2984] Error Code 4: Internal Error (inputT1: optimization profile
is missing values for shape input)
# 没有在运行时绑定形状张量的值
[TRT] [E] 3: [executionContext.cpp::resolveSlots::1481] Error Code 3: API Usage Error (Parameter check
failed at: runtime/api/executionContext.cpp::resolveSlots::1481, condition:
allInputShapesSpecified(routine)
)
# 绑定的形状张量的值与 profile 不匹配
[TRT] [E] 3: [executionContext.cpp::setInputShapeBinding::1016] Error Code 3: API Usage Error (Parameter
check failed at: runtime/api/executionContext.cpp::setInputShapeBinding::1016, condition: data[i] <=
profileMaxShape[i]. Supplied binding shapes [500,500,500,500] for bindings[3] exceed min ~ max range at
index 0, maximum shape in profile is 1, minimum shape in profile is 1, but supplied shape is 500.

)
```

dynamic shape 模式下的 shuffle + set_input

```
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()  # 需要使用 profile
config = builder.create_builder_config()
config.max_workspace_size = 1 << 30
inputT0 = network.add_input('inputT0', trt.float32, (-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)

oneLayer = network.add_constant([4], np.array([0, 1, 1, 1], dtype=np.int32))
shape0Layer = network.add_shape(inputT0)
shape1Layer = network.add_elementwise(shape0Layer.get_output(0), oneLayer.get_output(0),
trt.ElementWiseOperation.SUB)
sliceLayer = network.add_slice(inputT0, (0, 0, 0, 0), (0, 0, 0, 0), (1, 1, 1, 1))  # 给 inputT0 除了最高维
以外每一维减少 1
sliceLayer.set_input(2, shape1Layer.get_output(0))
#shape1 = [1] + [ x-1 for x in inputT0.shape[1:] ]
#sliceLayer = network.add_slice(inputT0,(0,0,0,0),shape1,(1,1,1,1))  # 错误的做法，因为 dynamic shape 模式下
inputT0.shape 可能含有 -1，不能作为新形状

network.mark_output(sliceLayer.get_output(0))
engineString = builder.build_serialized_network(network, config)
engine = trt.Runtime(logger).deserialize_cuda_engine(engineString)
context = engine.create_execution_context()
context.set_binding_shape(0, data.shape)
_, 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)  # 只打印形状
print(data)
print("outputH0:", outputH0.shape)
print(outputH0)

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

- 输出张量形状 (1,2,3,4)，结果与初始示例代码相同