IfCondition 结构

- 初始示例代码

初始示例代码

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

nIn0, cIn0, hIn0, wIn0 = 1, 3, 4, 5  # 输入张量 NCHW
data0 = np.arange(1, 1 + nIn0 * cIn0 * hIn0 * wIn0, dtype=np.float32).reshape(nIn0, cIn0, hIn0, wIn0)  # 输入数据
data1 = -data0

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
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (nIn0, cIn0, hIn0, wIn0))
#----------------------------------------------------------- --------------------# 替换部分
# 以 “inputT0.reshape(-1)[0] > 0” 作为判断条件
_H0 = network.add_slice(inputT0, [0, 0, 0, 0], [1, 1, 1, 1], [1, 1, 1, 1])
_H1 = network.add_reduce(_H0.get_output(0), trt.ReduceOperation.SUM, (1 << 0) + (1 << 1) + (1 << 2) + (1 << 3), False)
_H2 = network.add_identity(_H1.get_output(0))
_H2.set_output_type(0,trt.DataType.BOOL)
_H2.get_output(0).dtype = trt.DataType.BOOL

# 添加 condition 层
ifCondition = network.add_if_conditional()
ifConditionInputLayer = ifCondition.add_input(inputT0)
ifConditionConditionLayer = ifCondition.set_condition(_H2.get_output(0))  # 条件必须是 0 维 bool 型张量

# 判断条件成立时的分支
_H3 = network.add_elementwise(ifConditionInputLayer.get_output(0), inputT0, trt.ElementWiseOperation.SUM)

# 判断条件不成立时的分支
_H4 = network.add_unary(ifConditionInputLayer.get_output(0), trt.UnaryOperation.ABS)

# 标记 Condition 输出
ifConditionOutputLayer = ifCondition.add_output(_H3.get_output(0), _H4.get_output(0))
#----------------------------------------------------------- --------------------# 替换部分
network.mark_output(ifConditionOutputLayer.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()
```

```python
inputH0 = np.ascontiguousarray(data0.reshape(-1))  # 使用不同的输入数据以进入不同的分支
inputH1 = np.ascontiguousarray(data1.reshape(-1))
outputH0 = np.empty(context.get_binding_shape(1), dtype=trt.nptype(engine.get_binding_dtype(1)))
outputH1 = np.empty(context.get_binding_shape(1), dtype=trt.nptype(engine.get_binding_dtype(1)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, inputD1 = cudart.cudaMallocAsync(inputH1.nbytes, stream)
_, outputD0 = cudart.cudaMallocAsync(outputH0.nbytes, stream)
_, outputD1 = cudart.cudaMallocAsync(outputH1.nbytes, stream)

cudart.cudaMemcpyAsync(inputD0, inputH0.ctypes.data, inputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)  # 分两次推理，分别传入不同的数据
context.execute_async_v2([int(inputD0), int(outputD0)], stream)
cudart.cudaMemcpyAsync(outputH0.ctypes.data, outputD0, outputH0.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)

cudart.cudaMemcpyAsync(inputD1, inputH1.ctypes.data, inputH1.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyHostToDevice, stream)
context.execute_async_v2([int(inputD1), int(outputD1)], stream)
cudart.cudaMemcpyAsync(outputH1.ctypes.data, outputD1, outputH1.nbytes,
cudart.cudaMemcpyKind.cudaMemcpyDeviceToHost, stream)

cudart.cudaStreamSynchronize(stream)

print("inputH0 :", data0.shape)
print(data0)
print("outputH0:", outputH0.shape)
print(outputH0)
print("outputH1:", outputH1.shape)
print(outputH1)

cudart.cudaStreamDestroy(stream)
cudart.cudaFree(inputD0)
cudart.cudaFree(inputD1)
cudart.cudaFree(outputD0)
cudart.cudaFree(outputD1)
```

- 输入张量 data0 形状 (1,3,4,5)，data1 形状与 data0 相同，其数据为 data0 的相反数

$$\left[\left[\begin{bmatrix} 1. & 2. & 3. & 4. & 5. \\ 6. & 7. & 8. & 9. & 10. \\ 11. & 12. & 13. & 14. & 15. \\ 16. & 17. & 18. & 19. & 20. \end{bmatrix}\begin{bmatrix} 21. & 22. & 23. & 24. & 25. \\ 26. & 27. & 28. & 29. & 30. \\ 31. & 32. & 33. & 34. & 35. \\ 36. & 37. & 38. & 39. & 40. \end{bmatrix}\begin{bmatrix} 41. & 42. & 43. & 44. & 45. \\ 46. & 47. & 48. & 49. & 50. \\ 51. & 52. & 53. & 54. & 55. \\ 56. & 57. & 58. & 59. & 60. \end{bmatrix}\right]\right]$$

- 使用 data0 作为输入张量时，输出张量形状 (1,3,4,5)

$$\left[\left[\begin{bmatrix} 2. & 4. & 6. & 8. & 10. \\ 12. & 14. & 16. & 18. & 20. \\ 22. & 24. & 26. & 28. & 30. \\ 32. & 34. & 36. & 38. & 40. \end{bmatrix}\begin{bmatrix} 42. & 44. & 46. & 48. & 50. \\ 52. & 54. & 56. & 58. & 60. \\ 62. & 64. & 66. & 68. & 70. \\ 72. & 74. & 76. & 78. & 80. \end{bmatrix}\begin{bmatrix} 82. & 84. & 86. & 88. & 90. \\ 92. & 94. & 96. & 98. & 100. \\ 102. & 104. & 106. & 108. & 110. \\ 112. & 114. & 116. & 118. & 120. \end{bmatrix}\right]\right]$$

- 使用 data1 作为输入张量时，输出张量形状 (1,3,4,5)

$$\left[\left[\begin{bmatrix} 1. & 2. & 3. & 4. & 5. \\ 6. & 7. & 8. & 9. & 10. \\ 11. & 12. & 13. & 14. & 15. \\ 16. & 17. & 18. & 19. & 20. \end{bmatrix}\begin{bmatrix} 21. & 22. & 23. & 24. & 25. \\ 26. & 27. & 28. & 29. & 30. \\ 31. & 32. & 33. & 34. & 35. \\ 36. & 37. & 38. & 39. & 40. \end{bmatrix}\begin{bmatrix} 41. & 42. & 43. & 44. & 45. \\ 46. & 47. & 48. & 49. & 50. \\ 51. & 52. & 53. & 54. & 55. \\ 56. & 57. & 58. & 59. & 60. \end{bmatrix}\right]\right]$$

- 计算过程，等价于以下 python 代码

```python
if inputT0[0,0,0,0] > 0:
    return inputT0*2
else:
    return -inputT0
```

- IfCondition 结构的输出来自 IfConditionOutputLayer 层，实际上 IfConditionInputLayer 层和 IfConditionConditionLayer 层也提供了 get_output 方法，但是其输出张量被固定为 None