Select 层

- 初始示例代码

初始示例代码

```python
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
data0 = np.arange(nIn * cIn * hIn * wIn, dtype=np.float32).reshape(nIn, cIn, hIn, wIn)  # 输入数据
data1 = -data0
data2 = (np.arange(nIn * cIn * hIn * wIn) % 2).astype(np.int32).reshape(nIn, cIn, hIn, wIn)

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, (nIn, cIn, hIn, wIn))
inputT1 = network.add_input('inputT1', trt.DataType.FLOAT, (nIn, cIn, hIn, wIn))
inputT2 = network.add_input('inputT2', trt.DataType.INT32, (nIn, cIn, hIn, wIn))
#---------------------------------------------------------- --------------------# 替换部分
conditionLayer = network.add_identity(inputT2)  # 条件张量需要转化为 BOOL 类型
conditionLayer.set_output_type(0, trt.DataType.BOOL)
selectLayer = network.add_select(conditionLayer.get_output(0), inputT0, inputT1)
#---------------------------------------------------------- --------------------# 替换部分
network.mark_output(selectLayer.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(data0.reshape(-1))
inputH1 = np.ascontiguousarray(data1.reshape(-1))
inputH2 = np.ascontiguousarray(data2.reshape(-1))
outputH0 = np.empty(context.get_binding_shape(3), dtype=trt.nptype(engine.get_binding_dtype(3)))
_, inputD0 = cudart.cudaMallocAsync(inputH0.nbytes, stream)
_, inputD1 = cudart.cudaMallocAsync(inputH1.nbytes, stream)
_, inputD2 = cudart.cudaMallocAsync(inputH2.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)
context.execute_async_v2([int(inputD0), int(inputD1), int(inputD2), 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("outputH0:", outputH0.shape)
print(outputH0)

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

- 输入张量 0 形状 (1,3,4,5)，第 0 张量元素全正，第 1 张量的所有元素为第一个的相反数

$$\left[\left[\begin{bmatrix}0. & 1. & 2. & 3. & 4.\\ 5. & 6. & 7. & 8. & 9.\\ 10. & 11. & 12. & 13. & 14.\\ 15. & 16. & 17. & 18. & 19.\end{bmatrix}\begin{bmatrix}20. & 21. & 22. & 23. & 24.\\ 25. & 26. & 27. & 28. & 29.\\ 30. & 31. & 32. & 33. & 34.\\ 35. & 36. & 37. & 38. & 39.\end{bmatrix}\begin{bmatrix}40. & 41. & 42. & 43. & 44.\\ 45. & 46. & 47. & 48. & 49.\\ 50. & 51. & 52. & 53. & 54.\\ 55. & 56. & 57. & 58. & 59.\end{bmatrix}\right]\right]$$

- 输出张量形状 (1,3,4,5)，交替输出两个输入张量的值

$$\left[\left[\begin{bmatrix}-0. & 1. & -2. & 3. & -4.\\ 5. & -6. & 7. & -8. & 9.\\ -10. & 11. & -12. & 13. & -14.\\ 15. & -16. & 17. & -18. & 19.\end{bmatrix}\begin{bmatrix}-20. & 21. & -22. & 23. & -24.\\ 25. & -26. & 27. & -28. & 29.\\ -30. & 31. & -32. & 33. & -34.\\ 35. & -36. & 37. & -38. & 39.\end{bmatrix}\begin{bmatrix}-40. & 41. & -42. & 43. & -44.\\ 45. & -46. & 47. & -48. & 49.\\ -50. & 51. & -52. & 53. & -54.\\ 55. & -56. & 57. & -58. & 59.\end{bmatrix}\right]\right]$$