

LRN 层

- 初始示例代码
- window_size & alpha & beta & k

初始示例代码

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

nIn, cIn, hIn, wIn = 1, 3, 3, 3 # 输入张量 NCHW
data = np.tile(np.array([1, 2, 5], dtype=np.float32).reshape(cIn, 1, 1), (1, hIn, wIn)).reshape(nIn, cIn, hIn, wIn) # 输入数据.reshape(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()
inputT0 = network.add_input('inputT0', trt.DataType.FLOAT, (nIn, cIn, hIn, wIn))
#-----# 替换部分
lrnLayer = network.add_lrn(inputT0, 3, 1.0, 1.0, 0.0001) # LRN 窗口尺寸 n, 参数 alpha, beta, k
#-----# 替换部分
network.mark_output(lrnLayer.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 (1,3,3,3)

$$\left[\left[\begin{bmatrix} 1. & 1. & 1. \\ 1. & 1. & 1. \\ 1. & 1. & 1. \end{bmatrix} \begin{bmatrix} 2. & 2. & 2. \\ 2. & 2. & 2. \\ 2. & 2. & 2. \end{bmatrix} \begin{bmatrix} 5. & 5. & 5. \\ 5. & 5. & 5. \\ 5. & 5. & 5. \end{bmatrix} \right] \right]$$

- 输出张量形状 (1,3,3,3)

$$\left[\left[\begin{bmatrix} 0.59996396 & 0.59996396 & 0.59996396 \\ 0.59996396 & 0.59996396 & 0.59996396 \\ 0.59996396 & 0.59996396 & 0.59996396 \end{bmatrix} \begin{bmatrix} 0.19999799 & 0.19999799 & 0.19999799 \\ 0.19999799 & 0.19999799 & 0.19999799 \\ 0.19999799 & 0.19999799 & 0.19999799 \end{bmatrix} \begin{bmatrix} 0.51723605 & 0.51723605 & 0.51723605 \\ 0.51723605 & 0.51723605 & 0.51723605 \\ 0.51723605 & 0.51723605 & 0.51723605 \end{bmatrix} \right] \right]$$

- 计算过程: $n = 3, \alpha = \beta = 1., k = 0.0001$, 求和元素个数等于 $\lfloor \frac{n}{2} \rfloor$, 超出输入张量边界的部分按 0 计算

$$\frac{1^2}{(k + \frac{\alpha}{3}(0^2 + 1^2 + 2^2))^\beta} = 0.59996396,$$

$$\frac{2^2}{(k + \frac{\alpha}{3}(1^2 + 2^2 + 5^2))^\beta} = 0.19999799,$$

$$\frac{5^2}{(k + \frac{\alpha}{3}(2^2 + 5^2 + 0^2))^\beta} = 0.51723605$$

window_size & alpha & beta & k

```
lrnLayer          = network.add_lrn(inputT0, 3, 0.0, 2.0, 1.0)
#lrnLayer.window_size    = 3                                #
LRN 窗口尺寸, 范围 [3,15] 且为奇数
lrnLayer.alpha       = 1.0                                  #
alpha 值, 范围 [-1e20, 1e20]
lrnLayer.beta        = 1.0                                  #
beta 值, 范围 [0.01, 1e5f]
lrnLayer.k           = 0.0001                               # k
值, 范围 [1e-5, 1e10]
```

- 输出张量形状 (1,3,3,3), 结果与初始示例代码相同
- 使用 `window_size` 总是会报一个错误并且重设窗口尺寸无效

```
[TensorRT] ERROR: 3: [layers.h::setWindowSize::418] Error Code 3: Internal Error (Parameter check failed
at: /_src/build/cuda-11.3/8.2/x86_64/release/optimizer/api/layers.h::setWindowSize::418, condition:
(windowSize >= LRN_MIN_WINDOW && windowSize <= LRN_MAX_WINDOW && !(windowSize & 0x1))
)
```