paddingNd 层（padding 层）

- **括号中的层名和参数名适用于 TensorRT8 及之前版本，TensorRT9 及之后被废弃**
- 初始示例代码
- pre_padding_nd (pre_padding) & post_padding_nd (post_padding)
- 使用 paddingNd 层来进行 crop

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(1, 1 + nIn * cIn * hIn * wIn, dtype=np.float32).reshape(nIn, cIn, hIn, wIn)  #  输入数据

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))
#---------------------------------------------------------- --------------------# 替换部分
paddingLayer = network.add_padding_nd(inputT0, (1, 2), (3, 4))
#---------------------------------------------------------- --------------------# 替换部分
network.mark_output(paddingLayer.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)

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

- 输出张量形状 (1,3,8,11)，在输入张量的上、左、下、右分别垫起了 1、2、3、4 层元素 0

$$
\left[\left[\begin{matrix}\begin{bmatrix}0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1. & 2. & 3. & 4. & 5. & 0. & 0. & 0. & 0. \\ 0. & 0. & 6. & 7. & 8. & 9. & 10. & 0. & 0. & 0. & 0. \\ 0. & 0. & 11. & 12. & 13. & 14. & 15. & 0. & 0. & 0. & 0. \\ 0. & 0. & 16. & 17. & 18. & 19. & 20. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.\end{bmatrix} \\ \begin{bmatrix}0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 21. & 22. & 23. & 24. & 25. & 0. & 0. & 0. & 0. \\ 0. & 0. & 26. & 27. & 28. & 29. & 30. & 0. & 0. & 0. & 0. \\ 0. & 0. & 31. & 32. & 33. & 34. & 35. & 0. & 0. & 0. & 0. \\ 0. & 0. & 36. & 37. & 38. & 39. & 40. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.\end{bmatrix} \\ \begin{bmatrix}0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 41. & 42. & 43. & 44. & 45. & 0. & 0. & 0. & 0. \\ 0. & 0. & 46. & 47. & 48. & 49. & 50. & 0. & 0. & 0. & 0. \\ 0. & 0. & 51. & 52. & 53. & 54. & 55. & 0. & 0. & 0. & 0. \\ 0. & 0. & 56. & 57. & 58. & 59. & 60. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.\end{bmatrix}\end{matrix}\right]\right]
$$

- 使用旧版 API `add_padding` 会收到警告

```
DeprecationWarning: Use add_padding_nd instead.
```

- 要求输入张量维度不小于 3
- 仅支持 0 元素作为填充元素
- 仅支持输入张量的最内两维进行 padding

---

pre_padding_nd (pre_padding) & post_padding_nd (post_padding)

```python
paddingLayer = network.add_padding_nd(inputT0, (0, 0), (0, 0))
paddingLayer.pre_padding_nd = (1, 2)  # 重设上侧和左侧填充 0 层数
paddingLayer.post_padding_nd = (3, 4)  # 重设下侧和右侧填充 0 层数
```

- 输出张量形状 (1,3,8,11)，结果与初始示例代码相同
- 使用旧版 API `pre_padding` 或 `post_padding` 会收到警告

```
DeprecationWarning: Use pre_padding_nd instead.
DeprecationWarning: Use post_padding_nd instead.
```

- 目前 padding 只支持 2 维，使用其他维度的 padding 会收到报错

```
[TRT] [E] 3: [network.cpp::addPaddingNd::1243] Error Code 3: API Usage Error (Parameter check failed at:
optimizer/api/network.cpp::addPaddingNd::1243, condition: prePadding.nbDims == 2
)
```

使用 paddingNd 层来进行 crop

```
paddingLayer = network.add_padding_nd(inputT0, (-1, 0), (0, -2))
```

- padding 参数可以为负，输出张量尺寸 (1,3,3,3)，输入张量各 HW 维去掉了首行和末两列

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix} 6. & 7. & 8. \\ 11. & 12. & 13. \\ 16. & 17. & 18. \end{bmatrix} \\ \begin{bmatrix} 26. & 27. & 28. \\ 31. & 32. & 33. \\ 36. & 37. & 38. \end{bmatrix} \\ \begin{bmatrix} 46. & 47. & 48. \\ 51. & 52. & 53. \\ 56. & 57. & 58. \end{bmatrix}\end{bmatrix}\end{bmatrix}$$