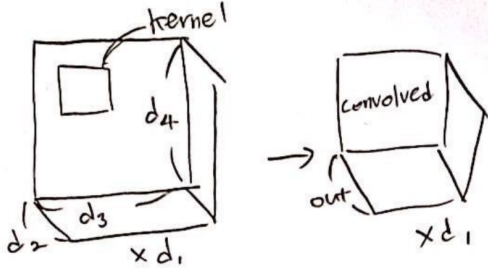


Convolutional Layers

torch.nn.Conv2d

`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, other_params..)(x)`



Require

- $|x| = (d_1, d_2, d_3, d_4)$ ($rank = 4$)
- $d_2 = in_channels$
- $d_3 + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0$
- $d_4 + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0$
- $groups|in_channels$ and $groups|out_channels$

Guarantees

- $|y| = (d_1, out_channels, h, w)$ where.. refers to the proof tree.

Comment

- Convolution layer입니다. 선배님의 자료를 pytorch의 사용에 맞게 풀어 쓴 것입니다.
- $kernel_size, stride$ 와 같은 옵션은 튜플로 구성될 수도 있습니다. (가로 세로에 대한 필터크기가 서로 다르도록) 이 경우를 위하여 proof 트리에서 $kernel_size[0], [1]$ 과 같은 표기를 사용하였습니다. 튜플이 아니라 스칼라 입력인 경우, $kernel_size[0], [1]$ 은 모두 $kernel_size$ 와 같습니다.
- 뒤의 $other_params..$ 부분은 텐서 shape에 전혀 영향을 주지 않는 인자입니다.

$\sigma \vdash E \Rightarrow e, c$

$$h = \left\lfloor \frac{e[3] + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1$$

$$w = \left\lfloor \frac{e[4] + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1$$

$e' = (e[1], out, h, w)$

$c_{dim} = \{(\mathbf{rank}(e) = 4) \wedge (e[2] = in)\}$

$c_h = \{(e[3] + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0)\}$

$c_w = \{(e[4] + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0)\}$

$c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0)\}$

$\sigma \vdash \text{Conv2d}(in, out, kernel_size, stride = 1, padding = 0, dilation = 1, groups = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_h \cup c_w \cup c_{group}$

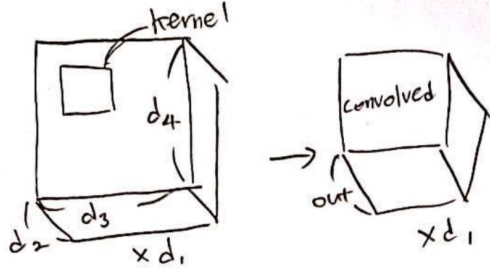
$kernel_size, stride, padding, dilation$ 는 가로-세로별 2-tuple로도 들어갈 수 있음

이 경우를 위해 $stride[0], stride[1]$ 으로 표기함

만일 $stride$ 가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 $[1]$ 은 $stride$ 값 자체를 의미

(Builtins) torch.conv2d, torch.nn.functional.conv2d

torch.conv2d(input, weight, bias=None, stride=1, padding=0, dilation=1, groups=1)



Require

- $|input| = (batch, in, h_{in}, w_{in})$ (rank = 4)
- $|weight| = (out, in_group, h_{weight}, w_{weight})$ (rank = 4)
- $bias$ is *None* or $|bias| = (out)$
- $h_{weight} + 2 \times padding[0] - dilation[0] \times (h_{weight} - 1) - 1 \geq 0$
- $w_{weight} + 2 \times padding[1] - dilation[1] \times (w_{weight} - 1) - 1 \geq 0$
- $groups|in, groups|out$ and $in_group \times group = in$

Guarantees

- $|y| = (batch, out, h, w)$ where.. refers to the proof tree.

Comment

- 컨볼루션 계산을 위해 사용하는 빌트인 함수입니다.
- torch.conv2d, torch.nn.functional.conv2d 모두 같은 방식으로 작동됩니다.

$$\sigma \vdash E \Rightarrow e, c$$

$$\sigma \vdash F \Rightarrow f, c$$

$$\sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None$$

$$(batch, in, h_{in}, w_{in}) = e$$

$$(out, in_group, h_{filter}, w_{filter}) = f$$

$$h = \left\lfloor \frac{h_{in} + 2 \times padding[0] - dilation[0] \times (h_{filter} - 1) - 1}{stride[0]} \right\rfloor + 1$$

$$w = \left\lfloor \frac{w_{in} + 2 \times padding[1] - dilation[1] \times (w_{filter} - 1) - 1}{stride[1]} \right\rfloor + 1$$

$$e' = (batch, out, h, w)$$

$$c_{dim} = \{(\mathbf{rank}(e) = 4) \wedge (\mathbf{rank}(f) = 4)\}$$

$$c_{bias} = \{((B = None) \vee (\mathbf{rank}(b) = 1 \wedge b[1] = out))\}$$

$$c_h = \{(h_{in} + 2 \times padding[0] - dilation[0] \times (h_{filter} - 1) - 1 \geq 0)\}$$

$$c_w = \{(w_{in} + 2 \times padding[1] - dilation[1] \times (w_{filter} - 1) - 1 \geq 0)\}$$

$$c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0) \wedge (in_group \times groups = in)\}$$

$$\sigma \vdash \text{conv2d}(E, F, B = None, stride = 1, padding = 0, dilation = 1, groups = 1) \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_h \cup c_w \cup c_{group}$$

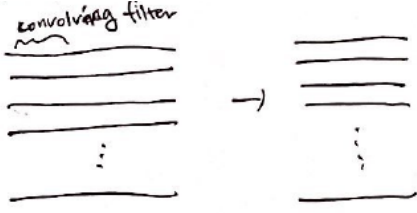
$kernel_size, stride, padding, dilation$ 는 가로-세로별 2-tuple로도 들어갈 수 있음

이 경우를 위해 $stride[0], stride[1]$ 으로 표기함

만일 $stride$ 가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 $[1]$ 은 $stride$ 값 자체를 의미

torch.nn.Conv1d

`torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, other_params..)(x)`

	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3)$ ($rank = 3$) • $d_2 = in_channels$ • $d_3 + 2 \times padding - dilation \times (kernel_size - 1) - 1 \geq 0$ • $groups in_channels$ and $groups out_channels$
	Guarantees
	<ul style="list-style-type: none"> • $y = (d_1, out_channels, w)$ where.. refers to the proof tree.
	Comment
	<ul style="list-style-type: none"> • Convolution 1차원 레이어입니다. • $kernel_size, stride$ 등은 1차원 튜플로 구성될 수도 있습니다.

$$\sigma \vdash E \Rightarrow e, c$$

$$w = \left\lfloor \frac{e[3] + 2 \times padding - dilation \times (kernel_size - 1) - 1}{stride} \right\rfloor + 1$$

$$e' = (e[1], out, w)$$

$$c_{dim} = \{(\mathbf{rank}(e) = 3) \wedge (e[2] = in)\}$$

$$c_w = \{(e[3] + 2 \times padding - dilation \times (kernel_size - 1) - 1 \geq 0)\}$$

$$c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0)\}$$

$$\sigma \vdash \text{Conv1d}(in, out, kernel_size, stride = 1, padding = 0, dilation = 1, groups = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_{group}$$

$kernel_size, stride, padding, dilation$ 는 1-length-tuple로 들어올 수 있음

(Builtins) torch.conv1d, torch.nn.functional.conv1d

$$\sigma \vdash E \Rightarrow e, c$$

$$\sigma \vdash F \Rightarrow f, c$$

$$\sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None$$

$$(batch, in, w_{in}) = e$$

$$(out, in_group, w_{filter}) = f$$

$$w = \left\lfloor \frac{w_{in} + 2 \times padding - dilation \times (w_{filter} - 1) - 1}{stride} \right\rfloor + 1$$

$$e' = (batch, out, w)$$

$$c_{dim} = \{(\mathbf{rank}(e) = 3) \wedge (\mathbf{rank}(f) = 3)\}$$

$$c_{bias} = \{((B = None) \vee (\mathbf{rank}(b) = 1 \wedge b[1] = out))\}$$

$$c_w = \{(w_{in} + 2 \times padding - dilation \times (w_{filter} - 1) - 1 \geq 0)\}$$

$$c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0) \wedge (in_group \times groups = in)\}$$

$$\sigma \vdash \text{conv1d}(E, F, B = None, stride = 1, padding = 0, dilation = 1, groups = 1) \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_w \cup c_{group}$$

$kernel_size, stride, padding, dilation$ 는 1-length-tuple로 들어올 수 있음

$$\begin{aligned}
& \sigma \vdash E \Rightarrow e, c \\
& z = \left\lfloor \frac{e[3] + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel_size}[0] - 1) - 1}{\text{stride}[0]} \right\rfloor + 1 \\
& h = \left\lfloor \frac{e[4] + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel_size}[1] - 1) - 1}{\text{stride}[1]} \right\rfloor + 1 \\
& w = \left\lfloor \frac{e[5] + 2 \times \text{padding}[2] - \text{dilation}[2] \times (\text{kernel_size}[2] - 1) - 1}{\text{stride}[2]} \right\rfloor + 1 \\
& e' = (e[1], \text{out}, z, h, w) \\
& c_{dim} = \{(\mathbf{rank}(e) = 5) \wedge (e[2] = \text{in})\} \\
& c_z = \{(e[3] + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel_size}[0] - 1) - 1 \geq 0)\} \\
& c_h = \{(e[4] + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel_size}[1] - 1) - 1 \geq 0)\} \\
& c_w = \{(e[5] + 2 \times \text{padding}[2] - \text{dilation}[2] \times (\text{kernel_size}[2] - 1) - 1 \geq 0)\} \\
& c_{group} = \{(\text{in}\%groups = 0) \wedge (\text{out}\%groups = 0)\} \\
\hline
& \sigma \vdash \text{Conv3d}(\text{in}, \text{out}, \text{kernel_size}, \text{stride} = 1, \text{padding} = 0, \text{dilation} = 1, \text{groups} = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_z \cup c_h \cup c_w \cup c_{group}
\end{aligned}$$

$\text{kernel_size}, \text{stride}, \text{padding}, \text{dilation}$ 는 깊이-가로-세로별 3-tuple로도 들어갈 수 있음
이 경우를 위해 $\text{stride}[0], [1], [2]$ 으로 표기함
만일 stride 가 튜플이 아닌 스칼라라면 $\text{stride}[0], [1]$ 또는 $[2]$ 는 stride 값 자체를 의미

(Builtins) torch.conv3d, torch.nn.functional.conv3d

$$\begin{aligned}
& \sigma \vdash E \Rightarrow e, c \\
& \sigma \vdash F \Rightarrow f, c \\
& \sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None \\
& (\text{batch}, \text{in}, z_{in}, h_{in}, w_{in}) = e \\
& (\text{out}, \text{in_group}, z_{filter}, h_{filter}, w_{filter}) = f \\
& z = \left\lfloor \frac{z_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (z_{filter} - 1) - 1}{\text{stride}[0]} \right\rfloor + 1 \\
& h = \left\lfloor \frac{h_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (h_{filter} - 1) - 1}{\text{stride}[1]} \right\rfloor + 1 \\
& w = \left\lfloor \frac{w_{in} + 2 \times \text{padding}[2] - \text{dilation}[2] \times (w_{filter} - 1) - 1}{\text{stride}[2]} \right\rfloor + 1 \\
& e' = (\text{batch}, \text{out}, z, h, w) \\
& c_{dim} = \{(\mathbf{rank}(e) = 5) \wedge (\mathbf{rank}(f) = 5)\} \\
& c_{bias} = \{((B = None) \vee (\mathbf{rank}(b) = 1 \wedge b[1] = \text{out}))\} \\
& c_z = \{(z_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (z_{filter} - 1) - 1 \geq 0)\} \\
& c_h = \{(h_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (h_{filter} - 1) - 1 \geq 0)\} \\
& c_w = \{(w_{in} + 2 \times \text{padding}[2] - \text{dilation}[2] \times (w_{filter} - 1) - 1 \geq 0)\} \\
& c_{group} = \{(\text{in}\%groups = 0) \wedge (\text{out}\%groups = 0) \wedge (\text{in_group} \times groups = \text{in})\} \\
\hline
& \sigma \vdash \text{conv2d}(E, F, B = None, \text{stride} = 1, \text{padding} = 0, \text{dilation} = 1, \text{groups} = 1) \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_z \cup c_h \cup c_w \cup c_{group}
\end{aligned}$$

$\text{kernel_size}, \text{stride}, \text{padding}, \text{dilation}$ 는 가로-세로별 3-tuple로도 들어갈 수 있음
이 경우를 위해 $\text{stride}[0], [1], [2]$ 으로 표기함
만일 stride 가 튜플이 아닌 스칼라라면 $\text{stride}[0], [1]$ 또는 $[2]$ 는 stride 값 자체를 의미

torch.nn.ConvTranspose1d

```
torch.nn.ConvTranspose1d(in_channels, out_channels, kernel_size, stride=1, padding=0,
output_padding=0, groups=1, bias=True, dilation=1, padding_mode='zeros')(x)
```

Require

- $|x| = (d_1, d_2, d_3)$ (rank = 3)
- $d_2 = \text{in_channels}$
- $\text{kernel_size} \leq d_3 + 2 \times \text{padding}$
- $\text{groups} | \text{in_channels}$ and $\text{groups} | \text{out_channels}$

Guarantees

- $|y| = (d_1, \text{out_channels}, w)$ where.. refers to the proof tree.

Comment

- Convolution에서 gradient를 구하기위한 레이어로 보시면 됩니다.
- kernel_size , stride 등은 1차원 튜플로 구성될 수도 있습니다.
- bias , pad_mode 옵션은 shape에 영향을 주지 않습니다.

$$\sigma \vdash E \Rightarrow e, c$$

$$w = (e[3] - 1) \times \text{stride} - 2 \times \text{pad} + \text{dilation} \times (\text{kernel} - 1) + \text{out_pad} + 1$$

$$e' = (e[1], \text{out}, w)$$

$$c_{\text{dim}} = \{(\text{rank}(e) = 3) \wedge (e[2] = \text{in}) \wedge (w > 0)\}$$

$$c_{\text{group}} = \{(\text{in} \% \text{groups} = 0) \wedge (\text{out} \% \text{groups} = 0)\}$$

$$\sigma \vdash \text{ConvTranspose1d}(\text{in}, \text{out}, \text{kernel}, \text{stride} = 1, \text{pad} = 0, \text{out_pad} = 0, \text{groups} = 1, \text{bias} = \text{True}, \text{dilation} = 1, \text{pad_mode})(E) \Rightarrow e', c \cup c_{\text{dim}} \cup c_{\text{group}}$$

$\text{kernel_size}, \text{stride}, \text{padding}, \text{dilation}$ 는 1-length-tuple로 들어올 수 있음

(Builtins) torch.conv_transpose1d, torch.nn.functional.conv_transpose1d

$$\sigma \vdash E \Rightarrow e, c_e$$

$$\sigma \vdash F \Rightarrow f, c_f$$

$$\sigma \vdash B \Rightarrow b, c_b \quad \text{if } B \text{ is not None}$$

$$w = (e[3] - 1) \times \text{stride} - 2 \times \text{pad} + \text{dilation} \times (f[3] - 1) + \text{out_pad} + 1$$

$$e' = (e[1], f[2] \times \text{groups}, w)$$

$$c_{\text{dim}} = \{(\text{rank}(e) = 3) \wedge (\text{rank}(f) = 3) \wedge (f[1] = e[2]) \wedge (w > 0)\}$$

$$c_{\text{bias}} = \{(B = \text{None} \vee b = (f[2] \times \text{groups}))\}$$

$$c_{\text{group}} = \{(\text{in} \% \text{groups} = 0)\}$$

$$\sigma \vdash \text{conv_transpose1d}(E, F, B = \text{None}, \text{stride} = 1, \text{pad} = 0, \text{out_pad} = 0, \text{groups} = 1, \text{dilation} = 1) \Rightarrow e', c \cup c_{\text{dim}} \cup c_{\text{bias}} \cup c_{\text{group}}$$

$\text{kernel_size}, \text{stride}, \text{padding}, \text{dilation}$ 는 1-length-tuple로 들어올 수 있음

Activations

torch.nn.MaxPool2d

torch.nn.MaxPool2d(kernel_size, stride=kernel_size, padding=0, dilation=1)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) • $d_3 + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0$ • $d_4 + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0$
	Guarantees
	<ul style="list-style-type: none"> • (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
	Comment
	<ul style="list-style-type: none"> • Convolution 다음 activation으로 자주 쓰이는 MaxPool 레이어 입니다.

$\sigma \vdash E \Rightarrow e, c$

$k = \text{rank}(e)$

$h_{orig} = e[k - 1]$

$w_{orig} = e[k]$

$h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1$

$w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1$

$e' = e[1:k - 2] @ (h, w)$

$c_{dim} = \{(k = 3 \vee k = 4)\}$

$c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0)\}$

$c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0)\}$

$\sigma \vdash \text{MaxPool2d}(kernel_size, stride = kernel_size, padding = 0, dilation = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h$

$kernel_size, stride, padding, dilation$ 는 가로-세로별 2-tuple로도 들어갈 수 있음

이 경우를 위해 $stride[0], stride[1]$ 으로 표기함

만일 $stride$ 가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 $[1]$ 은 $stride$ 값 자체를 의미

torch.nn.MaxPool2d(kernel_size, stride=..., dilation=1, return_indices=False, ceil_mode=False)(x)	
<p>return_indices가 True이면, 이전 인덱스로부터 왔는지 튜플로 반환 (같은 shape 두 개)</p>	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) • $d_3 + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0$ • $d_4 + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0$
	Guarantees
	<ul style="list-style-type: none"> • (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree. • $return_indices$가 True이면 인덱스 번호까지 튜플로 반환 • $ceil_mode$가 True이면 $floor$대신 $ceil$로 shape 계산

$$\begin{aligned}
& \sigma \vdash E \Rightarrow e, c \\
& k = \text{rank}(e) \\
& h_{orig} = e[k-1] \\
& w_{orig} = e[k] \\
& h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1 \\
& w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1 \\
& h_{ceil} = \left\lceil \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rceil + 1 \\
& w_{ceil} = \left\lceil \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rceil + 1 \\
& e' = \text{if } ceil_mode \text{ then } e[1:k-2] @ (h_{ceil}, w_{ceil}) \text{ else } e[1:k-2] @ (h, w) \\
& e_{out} = \text{if } return_indices \text{ then } (e', e') \text{ else } e' \\
& c_{dim} = \{(k = 3 \vee k = 4)\} \\
& c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0)\} \\
& c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0)\} \\
\hline
& \sigma \vdash \text{MaxPool2d}(kernel_size, stride, padding, dilation, return_indices, ceil_mode)(E) \\
& \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h
\end{aligned}$$

*return_indices*가 *True*이면 (결과, 인덱스) 튜플 형태로 반환

*ceil_mode*가 *True*이면 *floor*대신 *ceil*함수로 계산

$$\frac{\sigma \vdash \text{torch.nn.MaxPool2d}(E, other_params...) \Rightarrow e, c}{\sigma \vdash \text{max_pool2d}(E, other_params...) \Rightarrow e, c}$$

(Builtins) `torch.max_pool2d`나 `torch.nn.functional.max_pool2d`에 대한 적용

(Builtins) `torch.max_pool2d`, `torch.nn.functional.max_pool2d`

torch.max_pool2d(input, kernel_size, stride=kernel_size, padding=0, dilation=1, ceil_mode=False)	
	Require
	<ul style="list-style-type: none"> • $input = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) • $d_3 + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0$ • $d_4 + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0$
	Guarantees
	<ul style="list-style-type: none"> • (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
	Comment
	<ul style="list-style-type: none"> • <code>torch.max_pool2d</code> \equiv <code>torch.nn.functional.max_pool2d</code> • Builtin 함수인데, 특이한 점은 <code>return_indices</code> parameter가 없다는 것입니다. (<code>max_pool2d_with_indices</code>라는 다른 함수로 분리되어 있습니다.)

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
h_{orig} = e[k-1] \\
w_{orig} = e[k] \\
h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil) \\
w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil) \\
e' = e[1:k-2]@(h, w) \\
c_{dim} = \{(k = 3 \vee k = 4)\} \\
c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0)\} \\
c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0)\} \\
\hline
\sigma \vdash \text{max_pool2d}(E, kernel_size, stride = kernel_size, padding = 0, dilation = 1) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h
\end{array}$$

$kernel_size, stride, padding, dilation$ 는 가로-세로별 2-tuple로도 들어갈 수 있음
이 경우를 위해 $stride[0], stride[1]$ 으로 표기함
만일 $stride$ 가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 $[1]$ 은 $stride$ 값 자체를 의미

torch.nn.functional.max_pool2d_with_indices(input, kernel_size, stride=..., dilation=1, ceil_mode=False)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) • $kernel_size[0] \leq d_3 + 2 \times padding[0]$ • $kernel_size[1] \leq d_4 + 2 \times padding[1]$
	Guarantees
	<ul style="list-style-type: none"> • 2-tuple of (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
	Comment
	<ul style="list-style-type: none"> • torch.에는 없고, torch.nn.functional에만 있습니다.

$$\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \text{rank}(e) \\
h_{orig} = e[k-1] \\
w_{orig} = e[k] \\
h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil) \\
w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil) \\
e' = e[1:k-2]@(h, w) \\
c_{dim} = \{(k = 3 \vee k = 4)\} \\
c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel_size[0] - 1) - 1 \geq 0)\} \\
c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel_size[1] - 1) - 1 \geq 0)\} \\
\hline
\sigma \vdash \text{max_pool2d_with_indices}(E, kernel_size, stride, padding, dilation, ceil_mode) \\
\Rightarrow (e', e'), c \cup c_{dim} \cup c_w \cup c_h
\end{array}$$

torch.nn.AvgPool2d

torch.nn.AvgPool2d(kernel_size, stride=kernel_size, padding=0, other_params, ...)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) • $kernel_size[0] \leq d_3 + 2 \times padding[0]$ • $kernel_size[1] \leq d_4 + 2 \times padding[1]$
	Guarantees
	<ul style="list-style-type: none"> • (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
	Comment
	<ul style="list-style-type: none"> • 셀들의 최대값이 아닌 평균으로 정규화하는 레이어 • MaxPool2d와 비슷하나 <i>dilation</i>과 <i>return_indices</i> 옵션이 없음

$$\sigma \vdash E \Rightarrow e, c$$

$$k = \text{rank}(e)$$

$$h_{orig} = e[k - 1]$$

$$w_{orig} = e[k]$$

$$h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - kernel_size[0]}{stride[0]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil)$$

$$w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - kernel_size[1]}{stride[1]} \right\rfloor + 1 \quad (\text{if } ceil_mode \text{ is } True, \text{ then use } \lceil \cdot \rceil)$$

$$e' = e[1:k - 2] @ (h, w)$$

$$c_{dim} = \{(k = 3 \vee k = 4)\}$$

$$c_w = \{(kernel_size[0] \leq h_{orig} + 2 \times padding[0])\}$$

$$c_h = \{(kernel_size[1] \leq w_{orig} + 2 \times padding[1])\}$$

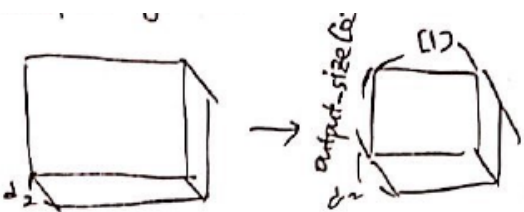
$$\sigma \vdash \text{AvgPool2d}(kernel_size, stride = kernel_size, padding = 0, other_params, \dots)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h$$

*kernel_size, stride, padding*는 가로-세로별 2-tuple로도 들어갈 수 있음

이 경우를 위해 *stride*[0], *stride*[1]으로 표기함

만일 *stride*가 튜플이 아닌 스칼라라면 *stride*[0] 또는 [1]은 *stride* 값 자체를 의미

torch.nn.AdaptiveAvgPool2d

torch.nn.AdaptiveAvgPool2d(output_size)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) – $\text{rank}(x) = 3 \text{ or } 4$
	Guarantees
	<ul style="list-style-type: none"> • $(d_1, d_2, output_size[0], output_size[1])$ or $(d_2, output_size[0], output_size[1])$
	Comment
	<ul style="list-style-type: none"> • 출력 shape를 강제하는 평균 pool 입니다. • <i>output_size</i>는 2-tuple이 될 수도 있습니다. • <i>output_size</i> > d_3 or d_4인 상황에도 오류없이 작동합니다.

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:k-2]@(output_size[0], output_size[1]) \\
c_{dim} = \{(k = 3 \vee k = 4)\} \\
\hline
\sigma \vdash \mathbf{AdaptiveAvgPool2d}(output_size)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h
\end{array}$$

$output_size$ 는 가로-세로별 2-tuple로도 들어갈 수 있음
이 경우를 위해 $output_size[0], output_size[1]$ 으로 표기함
만일 $output_size$ 가 튜플이 아닌 스칼라라면 $output_size[0]$ 또는 $[1]$ 은 $output_size$ 값 자체를 의미

torch.nn.AdaptiveAvgPool3d


torch.nn.AdaptiveAvgPool3d(output_size)(x)	
	Require
	<ul style="list-style-type: none"> $x = (d_1, d_2, d_3, d_4, d_5)$ or (d_2, d_3, d_4, d_5) $\mathbf{rank}(x) = 4$ or 5
	Guarantees
	<ul style="list-style-type: none"> $(d_1, d_2, d_3, output_size[0], output_size[1])$ or $(d_2, d_3, output_size[0], output_size[1])$
	Comment
	<ul style="list-style-type: none"> 출력 shape를 강제하는 평균 pool 입니다. $output_size$는 3-tuple이 될 수도 있습니다. $output_size > d_3, d_4$ or d_5인 상황에도 오류없이 작동합니다.

$$\begin{array}{c}
\sigma \vdash E \Rightarrow e, c \\
k = \mathbf{rank}(e) \\
e' = e[1:k-2]@(output_size[0], output_size[1]) \\
c_{dim} = \{(k = 4 \vee k = 5)\} \\
\hline
\sigma \vdash \mathbf{AdaptiveAvgPool3d}(output_size)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h
\end{array}$$

$output_size$ 는 깊이-가로-세로별 3-tuple로도 들어갈 수 있음
이 경우를 위해 $output_size[0], [1][2]$ 으로 표기함
만일 $output_size$ 가 튜플이 아닌 스칼라라면 $output_size[0], [1]$ 또는 $[2]$ 은 $output_size$ 값 자체를 의미

Normalizations

torch.nn.BatchNorm2d

torch.nn.BatchNorm2d(num_features, other_params...)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4)$ (rank = 4) • $d_2 = num_features$
	Guarantees
	<ul style="list-style-type: none"> • $y = (d_1, d_2, d_3, d_4)$ (same shape to x)

$$\sigma \vdash E \Rightarrow e, c$$

$$c' = \{(\mathbf{rank}(e) = 4) \wedge (e[2] = num_features)\}$$

$$\sigma \vdash \text{BatchNorm2d}(num_features, other_params)(E) \Rightarrow e, c \cup c'$$

torch.nn.BatchNorm3d

torch.nn.BatchNorm3d(num_features, other_params...)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3, d_4, d_5)$ (rank = 5) • $d_2 = num_features$
	Guarantees
	<ul style="list-style-type: none"> • $y = (d_1, d_2, d_3, d_4, d_5)$ (same shape to x)

$$\sigma \vdash E \Rightarrow e, c$$

$$c' = \{(\mathbf{rank}(e) = 5) \wedge (e[2] = num_features)\}$$

$$\sigma \vdash \text{BatchNorm3d}(num_features, other_params)(E) \Rightarrow e, c \cup c'$$

torch.nn.BatchNorm1d

torch.nn.BatchNorm1d(num_features, other_params...)(x)	
	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, d_3)$ (rank = 3) • $d_2 = num_features$
	Guarantees
	<ul style="list-style-type: none"> • $y = (d_1, d_2, d_3)$ (same shape to x)

$$\sigma \vdash E \Rightarrow e, c$$

$$c' = \{(\mathbf{rank}(e) = 3) \wedge (e[2] = num_features)\}$$

$$\sigma \vdash \text{BatchNorm1d}(num_features, other_params)(E) \Rightarrow e, c \cup c'$$

(Builtins) torch.batch_norm, torch.nn.functional.batch_norm

torch.batch_norm(input, running_mean, running_var, weight=None, bias=None, training=False, other_params,...)(x)	
	Require
	<ul style="list-style-type: none"> • $input = (d_1, d_2, d_3, \dots, d_k)$ • $k \geq 2$ • If <i>training</i> is <i>False</i> then <ul style="list-style-type: none"> – $running_mean = running_var = (d_2)$ Otherwise, <ul style="list-style-type: none"> – <i>running_mean</i> is <i>None</i> or $running_mean = (d_2)$ (also for <i>running_var</i>) • <i>weight</i> is <i>None</i> or $weight = (d_2)$ (also for <i>bias</i>)
	Guarantees
	<ul style="list-style-type: none"> • $y = (d_1, d_2, d_3, \dots, d_k) = x$
	Comment
	<ul style="list-style-type: none"> • BatchNormNd를 사용하기 위한 일반화된 함수입니다.

$$\sigma \vdash E \Rightarrow e, c_e$$

$$\sigma \vdash M \Rightarrow m, c_m \quad \text{if } M \text{ is not } None$$

$$\sigma \vdash V \Rightarrow v, c_v \quad \text{if } V \text{ is not } None$$

$$\sigma \vdash W \Rightarrow w, c_w \quad \text{if } W \text{ is not } None$$

$$\sigma \vdash B \Rightarrow b, c_b \quad \text{if } B \text{ is not } None$$

$$c_{rank} = \{\mathbf{rank}(e) \geq 2\}$$

$$c'_m = \{((training = True \wedge M = None) \vee (m = (d_2)))\}$$

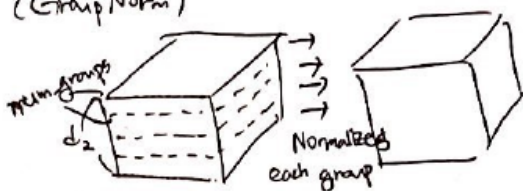
$$c'_v = \{((training = True \wedge V = None) \vee (v = (d_2)))\}$$

$$c'_w = \{((W = None) \vee (w = (d_2)))\}$$

$$c'_b = \{((B = None) \vee (b = (d_2)))\}$$

$$\sigma \vdash \text{batch_norm}(E, M, V, W, B, training, other_params, \dots) \Rightarrow e, c_e \cup c_m \cup \dots \cup c_b \cup c_{rank} \cup c'_m \cup \dots \cup c'_b$$

torch.nn.GroupNorm

torch.nn.GroupNorm(num_groups, num_channels, other_params, ...)(x)	
<p>(Group Norm)</p> 	Require
	<ul style="list-style-type: none"> • $x = (d_1, d_2, \dots, d_k)$ • $\mathbf{rank}(x) \geq 2$ • $num_groups num_channels$ • $d_2 = num_channels$
	Guarantees
	<ul style="list-style-type: none"> • $y = x$ (same shape)

$$\sigma \vdash E \Rightarrow e, c$$

$$c' = \{\mathbf{rank}(e) \geq 2 \wedge (e[2] \% num_groups = 0) \wedge (e[2] = num_channels)\}$$

$$\sigma \vdash \text{GroupNorm}(num_groups, num_channels, other_params, \dots)(E) \Rightarrow e, c \cup c'$$