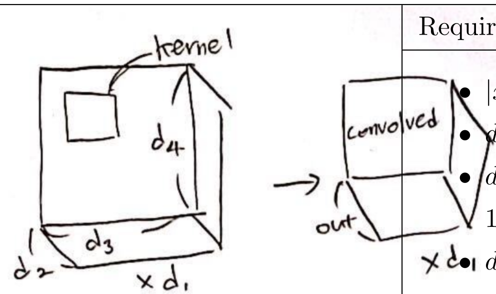
Convolutional Layers

`torch.nn.Conv2d`

<table>
<tr><td colspan="2"><code>torch.nn.Conv2d(in, out, kernel_size, stride=1, padding=0, dilation=1, groups=1)(x)</code></td></tr>
<tr><td rowspan="6">



</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, d_3, d_4) \quad (rank = 4)$
- $d_2 = in$
- $d_3 + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0$
- $d_4 + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0$
- $groups|in$ and $groups|out$

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

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

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- Convolution layer입니다. 선배님의 자료를 `pytorch`의 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.
- 추가적인 옵션을 더하여 `Conv2d(in, out, k, s, p, d, g, bias=True, padding_mode='zeros')`로 사용하기도 하지만, 뒤의 두 `bias`, `padding_mode`는 출력 shape에 아무런 영향을 주지 않습니다.

</td></tr>
</table>

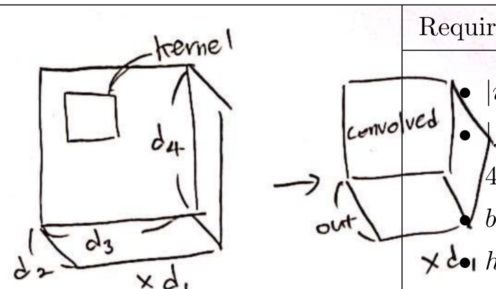
$$
\frac{\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
h = \left\lfloor \frac{e[3]+2\times padding[0]-dilation[0]\times(kernel\_size[0]-1)-1}{stride[0]} \right\rfloor + 1 \\
w = \left\lfloor \frac{e[4]+2\times padding[1]-dilation[1]\times(kernel\_size[1]-1)-1}{stride[1]} \right\rfloor + 1 \\
e' = (e[1], out, h, w) \\
c_{dim} = \{(\texttt{rank}(e) = 4) \wedge (e[2] = in)\} \\
c_h = \{(e[3] + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0)\} \\
c_w = \{(e[4] + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0)\} \\
c_{group} = \{(in\%groups = 0) \wedge (out\%groups = 0)\}
\end{array}}{\sigma \vdash \texttt{Conv2d}(in, out, kernel\_size, stride = 1, padding = 0, dilation = 1, groups = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_h \cup c_w \cup c_{gro}}
$$

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

이 경우를 위해 $stride[0], stride[1]$으로 표기

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

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

<table>
<tr><td colspan="2"><code>torch.conv2d(input, filter, bias=None, stride=1, padding=0, dilation=1, groups=1)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>

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

</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

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

</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

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

</td></tr>
</table>

$$
\frac{\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
\sigma \vdash F \Rightarrow f, c \\
\sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None \\
(batch, in, h_{in}, w_{in}) = e \\
(out, in\_group, h_{filter}, w_{filter}) = f \\
h = \left\lfloor \frac{h_{in} + 2\times padding[0] - dilation[0] \times (h_{filter} - 1) - 1}{stride[0]} \right\rfloor + 1 \\
w = \left\lfloor \frac{w_{in} + 2\times padding[1] - dilation[1] \times (w_{filter} - 1) - 1}{stride[1]} \right\rfloor + 1 \\
e' = (batch, out, h, w) \\
c_{dim} = \{(\texttt{rank}(e) = 4) \wedge (\texttt{rank}(f) = 4)\} \\
c_{bias} = \{((B = None) \vee (\texttt{rank}(b) = 1 \wedge b[1] = out))\} \\
c_h = \{(h_{in} + 2 \times padding[0] - dilation[0] \times (h_{filter} - 1) - 1 \geq 0)\} \\
c_w = \{(w_{in} + 2 \times padding[1] - dilation[1] \times (w_{filter} - 1) - 1 \geq 0)\} \\
c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0) \wedge (in\_group \times groups = in)\}
\end{array}}{\sigma \vdash \texttt{conv2d}(E, F, B = None, stride = 1, padding = 0, dilation = 1, groups = 1) \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_h \cup c_w \cup c_{gr}}
$$

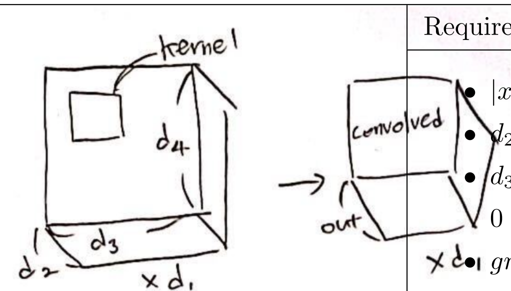
$kernel_size, stride, padding, dilation$는 가로-세로별 2-tuple로도 들어갈 수 있

이 경우를 위해 $stride[0], stride[1]$으로 표기

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

torch.nn.Conv1d

`torch.nn.Conv1d(in, out, kernel_size, stride=1, padding=0, dilation=1, groups=1)(x)`



Require

- $|x| = (d_1, d_2, d_3) \quad (rank = 3)$
- $d_2 = in$
- $d_3 + 2 \times padding - dilation \times (kernel\_size - 1) - 1 \geq 0$
- $groups | in$ and $groups | out$

Guarantees

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

Comment

- Convolution 1차원 레이어입니다.
- $kernel\_size$, $stride$ 등은 1차원 튜플로 구성될 수도 있습니다.
- 추가적인 옵션을 더하여 `Conv1d(in, out, k, s, p, d, g, bias=True, padding_mode='zeros')`로 사용하기도 하지만, 뒤의 두 `bias`, `padding_mode`는 출력 shape에 아무런 영향을 주지 않습니다.

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ w = \left\lfloor \frac{e[3] + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} \right\rfloor + 1 \\ e' = (e[1], out, w) \\ c_{dim} = \{(\texttt{rank}(e) = 3) \wedge (e[2] = in)\} \\ c_w = \{(e[3] + 2 \times padding - dilation \times (kernel\_size - 1) - 1 \geq 0)\} \\ c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0)\}\end{array}}{\sigma \vdash \texttt{Conv1d}(in, out, kernel\_size, stride = 1, padding = 0, dilation = 1, groups = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_{group}}$$

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

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

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ \sigma \vdash F \Rightarrow f, c \\ \sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None \\ (batch, in, w_{in}) = e \\ (out, in\_group, w_{filter}) = f \\ w = \left\lfloor \frac{w_{in} + 2 \times padding - dilation \times (w_{filter} - 1) - 1}{stride} \right\rfloor + 1 \\ e' = (batch, out, w) \\ c_{dim} = \{(\texttt{rank}(e) = 3) \wedge (\texttt{rank}(f) = 3)\} \\ c_{bias} = \{((B = None) \vee (\texttt{rank}(b) = 1 \wedge b[1] = out))\} \\ c_w = \{(w_{in} + 2 \times padding - dilation \times (w_{filter} - 1) - 1 \geq 0)\} \\ c_{group} = \{(in \% groups = 0) \wedge (out \% groups = 0) \wedge (in\_group \times groups = in)\}\end{array}}{\sigma \vdash \texttt{conv1d}(E, F, B = None, stride = 1, padding = 0, dilation = 1, groups = 1) \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_w \cup c_{group}}$$

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

`torch.nn.Conv3d`

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

$kernel_size, stride, padding, dilation$는 깊이-가로-세로별 3-tuple로도 들어갈

이 경우를 위해 $stride[0], [1], [2]$으로

만일 $stride$가 튜플이 아닌 스칼라라면 $stride[0], [1]$ 또는 $[2]$는 $stride$ 값 자체

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

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

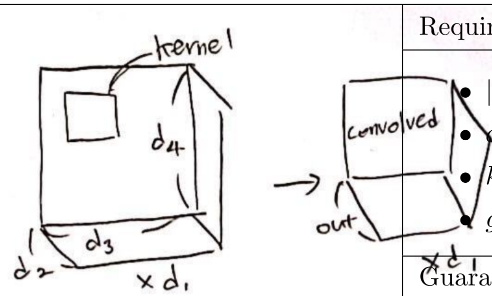
$kernel_size, stride, padding, dilation$는 가로-세로별 3-tuple로도 들어갈

이 경우를 위해 $stride[0], [1], [2]$으로

만일 $stride$가 튜플이 아닌 스칼라라면 $stride[0], [1]$ 또는 $[2]$는 $stride$ 값 자체

`torch.nn.ConvTranspose1d`

`torch.nn.Conv1d(in, out, kernel, stride=1, pad=0, out_pad=0, groups=1, bias=True, dilation=1, pa`



Require

- $|x| = (d_1, d_2, d_3) \quad (rank = 3)$
- $d_2 = in$
- $kernel \leq d_3 + 2 \times padding$
- $groups|in$ and $groups|out$

Guarantees

- $|y| = (d_1, out, w)$ 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.

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ w = (e[3]-1) \times stride - 2 \times pad + dilation \times (kernel - 1) + out\_pad + 1 \\ e' = (e[1], out, w) \\ c_{dim} = \{(\texttt{rank}(e) = 3) \land (e[2] = in) \land (w > 0)\} \\ c_{group} = \{(in \% groups = 0) \land (out \% groups = 0)\}\end{array}}{\begin{array}{l}\sigma \vdash \texttt{ConvTranspose1d}(in, out, kernel, stride = 1, pad = 0, out\_pad = 0, groups = 1, bias = True, dilation = 1, pad \\ \Rightarrow e', c \cup c_{dim} \cup c_{group}\end{array}}$$

$kernel_size, stride, padding, dilation$는 1-length-tuple로 들

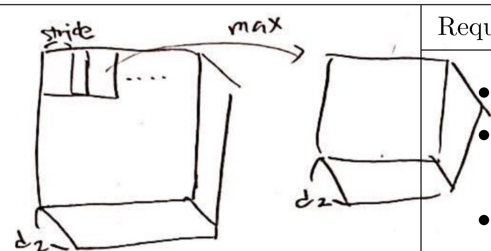
(Builtins) `torch.conv_transpose1d`, `torch.nn.functional.conv_transpose1d`

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c_e \\ \sigma \vdash F \Rightarrow f, c_f \\ \sigma \vdash B \Rightarrow b, c_b \quad \text{if } B \text{ is not } None \\ w = (e[3]-1) \times stride - 2 \times pad + dilation \times (f[3]-1) + out\_pad + 1 \\ e' = (e[1], f[2] \times groups, w) \\ c_{dim} = \{(\texttt{rank}(e) = 3) \wedge (\texttt{rank}(f) = 3) \wedge (f[1] = e[2]) \wedge (w > 0)\} \\ c_{bias} = \{(B = None \vee b = (f[2] \times groups))\} \\ c_{group} = \{(in\%groups = 0)\} \end{array}}{\begin{array}{l} \sigma \vdash \texttt{conv\_transpose1d}(E, F, B = None, stride = 1, pad = 0, out\_pad = 0, groups = 1, dilation = 1) \\ \Rightarrow e', c \cup c_{dim} \cup c_{bias} \cup c_{group} \end{array}}$$

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

Activations

`torch.nn.MaxPool2d`

<table>
<tr><td colspan="2"><code>torch.nn.MaxPool2d(kernel_size, stride=kernel_size, padding=0, dilation=1)(x)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>$|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) $d_3 + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0$ $d_4 + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>(d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.</td></tr>
<tr><td>Comment</td></tr>
<tr><td>Convolution 다음 activation으로 자주 쓰이는 MaxPool 레이어 입니다.</td></tr>
</table>

$$
\frac{\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
k = \texttt{rank}(e) \\
h_{orig} = e[k-1] \\
w_{orig} = e[k] \\
h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0]-1) - 1}{stride[0]} \right\rfloor + 1 \\
w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1]-1) - 1}{stride[1]} \right\rfloor + 1 \\
e' = e[1{:}k-2] @ (h, w) \\
c_{dim} = \{(k = 3 \vee k = 4)\} \\
c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0)\} \\
c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0)\}
\end{array}}{\sigma \vdash \texttt{MaxPool2d}(kernel\_size, stride = kernel\_size, padding = 0, dilation = 1)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h}
$$

$kernel_size, stride, padding, dilation$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

<table>
<tr><td colspan="2"><code>torch.nn.MaxPool2d(kernel_size, stride=..., dilation=1, return_indices=False, ceil_mode=False)(x</code></td></tr>
<tr><td rowspan="4">return_indices가 True이면,
어떤 인덱스로부터 왔는지
튜플로 반환 (같은 shape 두 개)</td><td>Require</td></tr>
<tr><td>$|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) $d_3 + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0$ $d_4 + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>(d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
$return_indices$가 $True$이면 인덱스 번호까지 튜플로 반환
$ceil_mode$가 $True$이면 $floor$대신 $ceil$로 shape 계산</td></tr>
</table>

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

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

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

$$
\frac{\sigma \vdash \texttt{torch.nn.MaxPool2d}(E, other\_params...) \Rightarrow e, c}{\sigma \vdash \texttt{max\_pool2d}(E, other\_params...) \Rightarrow e, c}
$$

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

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

<table>
<tr><td colspan="2"><code>torch.max_pool2d(x, kernel_size, stride=kernel_size, padding=0, dilation=1, ceil_mode=False)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>$|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4) $d_3 + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0$ $d_4 + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>(d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.</td></tr>
<tr><td>Comment</td></tr>
<tr><td><code>torch.max_pool2d</code> ≡ <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 분리되어있습니다.)</td></tr>
</table>

$$
\frac{
\begin{array}{ll}
\sigma \vdash E \Rightarrow e, c & \\
k = \texttt{rank}(e) & \\
h_{orig} = e[k-1] & \\
w_{orig} = e[k] & \\
h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1 & \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil \cdot \rceil) \\
w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1}{stride[1]} \right\rfloor + 1 & \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil \cdot \rceil) \\
e' = e[1{:}k-2]@(h, w) & \\
c_{dim} = \{(k = 3 \vee k = 4)\} & \\
c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1 \geq 0)\} & \\
c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1] - 1) - 1 \geq 0)\} &
\end{array}
}{
\sigma \vdash \texttt{max\_pool2d}(E, kernel\_size, stride = kernel\_size, padding = 0, dilation = 1) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h
}
$$

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

이 경우를 위해 $stride[0], stride[1]$으로 표기함

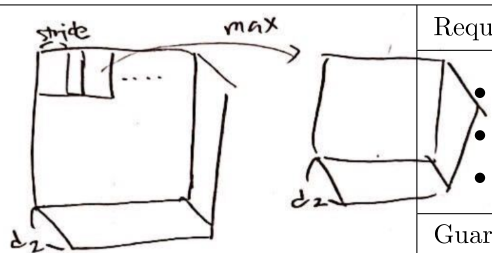
만일 $stride$가 튜플이 아닌 스칼라라면 $stride[0]$ 또는 $[1]$은 $stride$ 값 자체를 의미

<table>
<tr><td colspan="2"><code>torch.nn.functional.max_pool2d_with_indices(x, kernel_size, stride=..., dilation=1, ceil_mode=Fal</code></td></tr>
<tr><td rowspan="6">return_indices가 True이면,
어떤 인덱스로부터 왔는지
튜플로 반환 (같은 shape 두 개)</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4)
• $kernel\_size[0] \leq d_3 + 2 \times padding[0]$
• $kernel\_size[1] \leq d_4 + 2 \times padding[1]$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• 2-tuple of (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• <code>torch.</code>에는 없고, <code>torch.nn.functional</code>에만 있습니다.</td></tr>
</table>

$$
\begin{aligned}
&\sigma \vdash E \Rightarrow e, c\\
&k = \texttt{rank}(e)\\
&h_{orig} = e[k-1]\\
&w_{orig} = e[k]\\
&h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0]-1) - 1}{stride[0]} \right\rfloor + 1 && \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil\cdot\rceil)\\
&w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1]-1) - 1}{stride[1]} \right\rfloor + 1 && \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil\cdot\rceil)\\
&e' = e[1{:}k-2]@(h, w)\\
&c_{dim} = \{(k = 3 \vee k = 4)\}\\
&c_h = \{(h_{orig} + 2 \times padding[0] - dilation[0] \times (kernel\_size[0]-1) - 1 \geq 0)\}\\
&c_w = \{(w_{orig} + 2 \times padding[1] - dilation[1] \times (kernel\_size[1]-1) - 1 \geq 0)\}
\end{aligned}
$$

$$
\frac{}{\sigma \vdash \texttt{max\_pool2d\_with\_indices}(E, kernel\_size, stride, padding, dilation, ceil\_mode) \Rightarrow (e', e'), c \cup c_{dim} \cup c_w \cup c_h}
$$

`torch.nn.AvgPool2d`

<table>
<tr><td colspan="2"><code>torch.nn.AvgPool2d(kernel_size, stride=kernel_size, padding=0, other_params, ...)(x)</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4)
• $kernel\_size[0] \leq d_3 + 2 \times padding[0]$
• $kernel\_size[1] \leq d_4 + 2 \times padding[1]$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 셀들의 평균으로 정규화하는 레이어
• <code>MaxPool2d</code>와 비슷하나 dilation과 return_indices 옵션이 없음</td></tr>
</table>

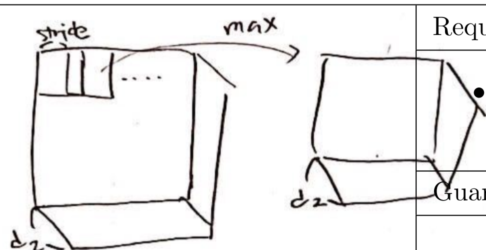
$$
\frac{\begin{array}{ll}
\sigma \vdash E \Rightarrow e, c & \\
k = \texttt{rank}(e) & \\
h_{orig} = e[k-1] & \\
w_{orig} = e[k] & \\
h = \left\lfloor \frac{h_{orig} + 2 \times padding[0] - kernel\_size[0]}{stride[0]} \right\rfloor + 1 & \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil \cdot \rceil\text{)} \\
w = \left\lfloor \frac{w_{orig} + 2 \times padding[1] - kernel\_size[1]}{stride[1]} \right\rfloor + 1 & \text{(if } ceil\_mode \text{ is } True\text{, then use } \lceil \cdot \rceil\text{)} \\
e' = e[1{:}k-2]@(h, w) & \\
c_{dim} = \{(k = 3 \vee k = 4)\} & \\
c_w = \{(kernel\_size[0] \leq h_{orig} + 2 \times padding[0])\} & \\
c_h = \{(kernel\_size[1] \leq w_{orig} + 2 \times padding[1])\} &
\end{array}}{\sigma \vdash \texttt{MaxPool2d}(kernel\_size, stride = kernel\_size, padding = 0, other\_params, ...)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h}
$$

$kernel_size, stride, padding$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

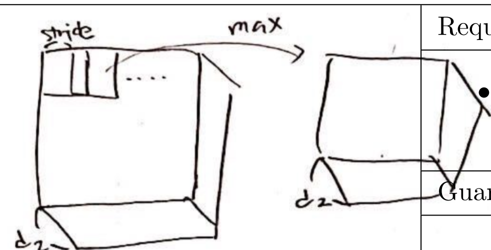
torch.nn.AdaptiveAvgPool2d

<table>
<tr><td colspan="2"><code>torch.nn.AdaptiveAvgPool2d(output_size)(x)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4)$ or (d_2, d_3, d_4)
– $\texttt{rank}(|x|) = 3$ or 4</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $(d_1, d_2, output\_size[0], output\_size[1])$ or $(d_2, output\_size[0], output\_size[1])$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 출력 shape를 강제하는 평균 pool 입니다.
• $output_size$는 2-tuple이 될 수도 있습니다.
• $output\_size > d_3$ or d_4인 상황에도 오류없이 작동합니다.</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}k-2]@(output\_size[0], output\_size[1]) \\ c_{dim} = \{(k = 3 \lor k = 4)\}\end{array}}{\sigma \vdash \texttt{AdaptiveAvgPool2d}(output\_size)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h}$$

*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`

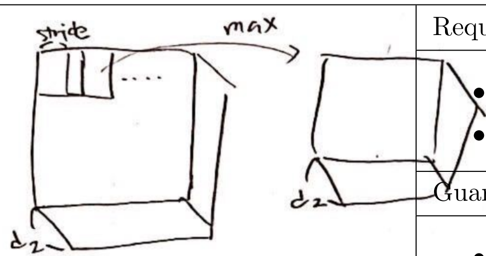
<table>
<tr><td colspan="2"><code>torch.nn.AdaptiveAvgPool3d(output_size)(x)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4, d_5)$ or (d_2, d_3, d_4, d_5)
– $\texttt{rank}(|x|) = 4$ or 5</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $(d_1, d_2, d_3, output\_size[0], output\_size[1])$
or $(d_2, d_3, output\_size[0], output\_size[1])$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 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.</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}k-2]@(output\_size[0], output\_size[1]) \\ c_{dim} = \{(k = 4 \lor k = 5)\}\end{array}}{\sigma \vdash \texttt{AdaptiveAvgPool3d}(output\_size)(E) \Rightarrow e', c \cup c_{dim} \cup c_w \cup c_h}$$

*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

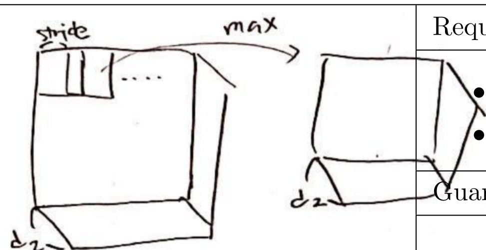
Normalizations

torch.nn.BatchNorm2d

<table>
<tr><td colspan="2"><code>torch.nn.BatchNorm2d(num_features, other_params...)(x)</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4)$ (<code>rank</code> $= 4$)
• $d_2 = num\_features$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = (d_1, d_2, d_3, d_4)$ (same shape to x)</td></tr>
</table>

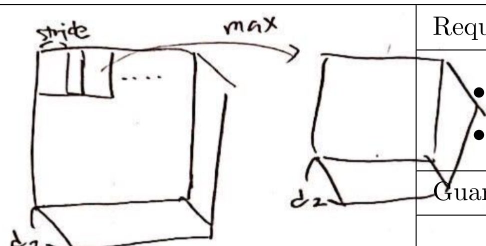
$$\frac{\sigma \vdash E \Rightarrow e, c \qquad c' = \{(\texttt{rank}(e) = 4) \wedge (e[2] = num\_features)\}}{\sigma \vdash \texttt{BatchNorm2d}(num\_features, other\_params)(E) \Rightarrow e, c \cup c'}$$

torch.nn.BatchNorm3d

<table>
<tr><td colspan="2"><code>torch.nn.BatchNorm3d(num_features, other_params...)(x)</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3, d_4, d_5)$ (<code>rank</code> $= 5$)
• $d_2 = num\_features$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = (d_1, d_2, d_3, d_4, d_5)$ (same shape to x)</td></tr>
</table>

$$\frac{\sigma \vdash E \Rightarrow e, c \qquad c' = \{(\texttt{rank}(e) = 5) \wedge (e[2] = num\_features)\}}{\sigma \vdash \texttt{BatchNorm3d}(num\_features, other\_params)(E) \Rightarrow e, c \cup c'}$$

torch.nn.BatchNorm1d

<table>
<tr><td colspan="2"><code>torch.nn.BatchNorm1d(num_features, other_params...)(x)</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>• $|x| = (d_1, d_2, d_3)$ (<code>rank</code> $= 3$)
• $d_2 = num\_features$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = (d_1, d_2, d_3)$ (same shape to x)</td></tr>
</table>

$$\frac{\sigma \vdash E \Rightarrow e, c \qquad c' = \{(\texttt{rank}(e) = 3) \wedge (e[2] = num\_features)\}}{\sigma \vdash \texttt{BatchNorm1d}(num\_features, other\_params)(E) \Rightarrow e, c \cup c'}$$

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

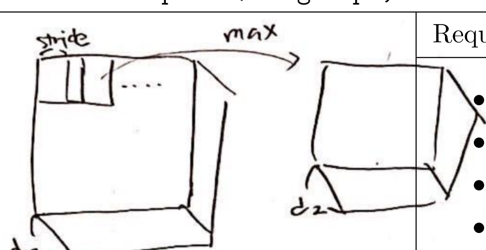
`torch.batch_norm(x, mean, var, weight=None, bias=None, training=False, other_params,...)(x)`	
	Require

	• $\|x\| = (d_1, d_2, d_3, \ldots, d_k)$
	• $k \geq 2$
	• If *training* is *False* then
	  – $\|mean\| = \|var\| = (d_2)$
	 Otherwise,
	  – *mean* is *None* or $\|mean\| = (d_2)$ (also for *var*)
	• *weight* is *None* or $\|weight\| = (d_2)$ (also for *bias*)

	Guarantees
	• $\|y\| = (d_1, d_2, d_3, \ldots, d_k) = \|x\|$
	Comment
	• `BatchNormNd`를 사용하기 위한 일반화된 함수입니다.

$$\frac{\begin{array}{rl} \sigma \vdash E \Rightarrow e, c_e & \\ \sigma \vdash M \Rightarrow m, c_m & \text{if } M \text{ is not } None \\ \sigma \vdash V \Rightarrow v, c_v & \text{if } V \text{ is not } None \\ \sigma \vdash W \Rightarrow w, c_w & \text{if } W \text{ is not } None \\ \sigma \vdash B \Rightarrow b, c_b & \text{if } B \text{ is not } None \\ c_{rank} = \{(\texttt{rank}(e) \geq 2)\} & \\ c'_m = \{((training = True \wedge M = None) \vee (m = (d_2))\} & \\ c'_v = \{((training = True \wedge V = None) \vee (v = (d_2))\} & \\ c'_w = \{((W = None) \vee (w = (d_2))\} & \\ c'_b = \{((B = None) \vee (b = (d_2))\} & \end{array}}{\sigma \vdash \texttt{batch\_norm}(E, M, V, W, B, training, other\_params, ...) \Rightarrow e, c_e \cup c_m \cup \cdots \cup c_b \cup c_{rank} \cup c'_m \cup \cdots \cup c'_b}$$

`torch.nn.GroupNorm`

`torch.nn.GroupNorm(num_groups, num_channels, other_params, ...)(x)`	
	Require
	• $\|x\| = (d_1, d_2, \ldots, d_k)$ • $\texttt{rank}(\|x\|) \geq 2$ • $num\_groups \| num\_channels$ • $d_2 = num\_channels$
	Guarantees
	• $\|y\| = \|x\|$ (same shape)

$$\frac{\sigma \vdash E \Rightarrow e, c \qquad c' = \{(\texttt{rank}(e) \geq 2) \wedge (e[2] \% num\_groups = 0) \wedge (e[2] = num\_channels)\}}{\sigma \vdash \texttt{GroupNorm}(num\_groups, num\_channels, other\_params, ...)(E) \Rightarrow e, c \cup c'}$$