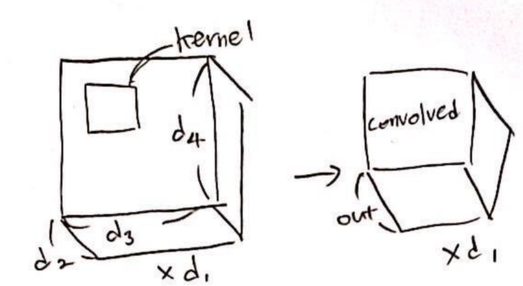
Convolutional Layers

torch.nn.Conv2d

`torch.nn.Conv2d(in, out, kernel_size, stride=1, padding=0, dilation=1, groups=1)(x)`



Require

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

Guarantees

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

Comment

- 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.

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

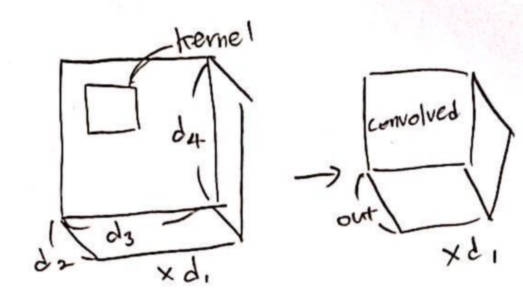
$kernel_size, stride, padding, dilation$는 가로-세로별 2-tuple로도 들어갈 수 있음

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, filter, bias=None, stride=1, padding=0, dilation=1, groups=1)
```



Require

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

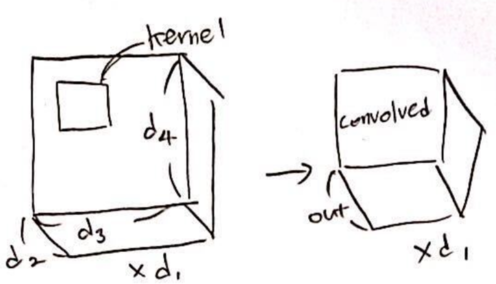
$$
\frac{
\begin{array}{l}
\sigma \vdash E \Rightarrow e, c \\
\sigma \vdash F \Rightarrow f, c \\
\sigma \vdash B \Rightarrow b, c \quad \text{if } B \text{ is not } None \\
(batch, in, h_{in}, w_{in}) = e \\
(out, in\_group, h_{filter}, w_{filter}) = f \\
h = \left\lfloor \frac{h_{in} + 2 \times padding[0]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

$kernel_size, stride, padding, dilation$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`

<table>
<tr><td colspan="2"><code>torch.nn.Conv1d(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>

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

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

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

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

- 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.

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

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

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

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

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

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

`torch.nn.Conv3d`

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

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

이 경우를 위해 $stride[0], [1], [2]$으로 표기함

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

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

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

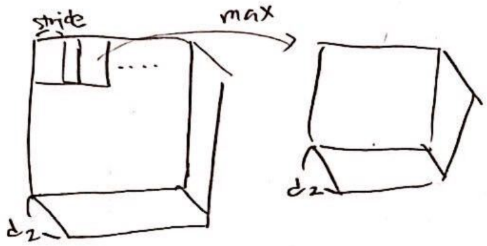
$kernel_size, stride, padding, dilation$는 가로-세로별 3-tuple로도 들어갈 수 있음

이 경우를 위해 $stride[0], [1], [2]$으로 표기함

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

Activations

`torch.nn.MaxPool2d`

<table>
<tr><td colspan="2"><code>torch.nn.MaxPool2d(kernel_size, stride=kernel_size, padding=0, dilation=1)</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>

- Convolution 다음 activation으로 자주 쓰이는 MaxPool 레이어 입니다.

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

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

$kernel_size, stride, padding, dilation$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)</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)
- $kernel\_size[0] \leq d_3 + 2 \times padding[0]$
- $kernel\_size[1] \leq d_4 + 2 \times padding[1]$

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

- (d_1, d_2, h, w) or (d_2, h, w) where.. proof tree.
- $return_indices$가 $True$이면 인덱스 번호까지 튜플로 반환
- $ceil_mode$가 $True$이면 $floor$대신 $ceil$로 shape 계산

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

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

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

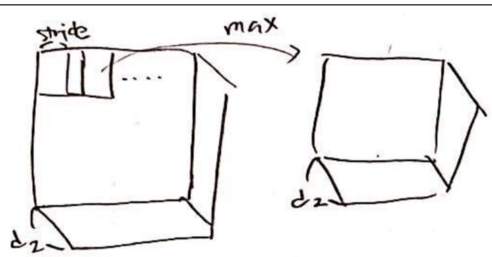
$ceil_mode$가 $True$이면 $floor$대신 $ceil$함수로 계산

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

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

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>

- $|x| = (d_1, d_2, d_3, d_4)$ ($\texttt{rank} = 4$)
- $d_2 = num\_features$

</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>

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