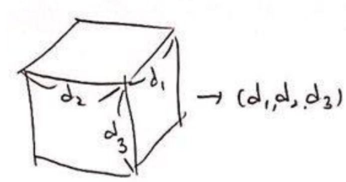
Matmul Layers

Activations

Technique

Wrapper

`torch.Tensor.size`

<table>
<tr><td colspan="2"><code>a.size()</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>

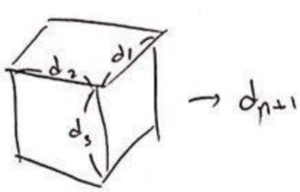
- $|\mathsf{a}| = (d_1, d_2, \ldots, d_k)$

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

- $(d_1, d_2, \ldots, d_k)$를 튜플로 반환

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

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\texttt{size}() \Rightarrow \mathit{shapeToTuple}(e), c}$$

<table>
<tr><td colspan="2"><code>a.size(n)</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>

- $|\mathsf{a}| = (d_1, d_2, \ldots, d_k)$
- $0 \leq n < k$

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

- d_{n+1}을 숫자(int)로 반환

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

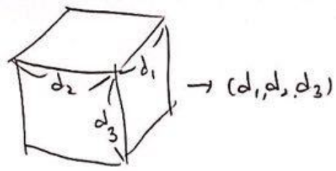
$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash E.\texttt{size}(n) \Rightarrow e[n+1], c \cup c'}$$

`torch.tensor`

<table>
<tr><td colspan="2"><code>torch.tensor(x)</code></td></tr>
<tr><td rowspan="2"></td><td>Comment</td></tr>
<tr><td><code>numpy</code>나 파이썬 리스트로 선언된 객체를 <code>torch</code>에서 호환가능한 형태로 바꾸는 함수.</td></tr>
</table>

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \texttt{tensor}(E) \Rightarrow e, c}$$

`torch.Tensor.shape`

<table>
<tr><td colspan="2"><code>a.shape</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>• $|\mathsf{a}| = (d_1, d_2, \ldots, d_k)$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $(d_1, d_2, \ldots, d_k)$를 튜플로 반환</td></tr>
</table>

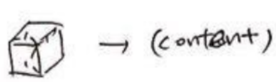
$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash E.\mathtt{shape} \Rightarrow \mathit{shapeToTuple}(e), c}$$

`torch.range`

<table>
<tr><td colspan="2"><code>torch.range(s, e, d)</code></td></tr>
<tr><td rowspan="6">$(s, s+d, s+2d, \cdots, s+\lfloor \frac{e-s}{d} \rfloor d)$</td><td>Require</td></tr>
<tr><td>• $d \neq 0$
• $(e-s)/d > 0$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = (1 + (e-s)/d)$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• $(s, s+d, s+2d, \ldots)$를 반환
• 기본값은 $s = 0,\ d = 1$</td></tr>
</table>

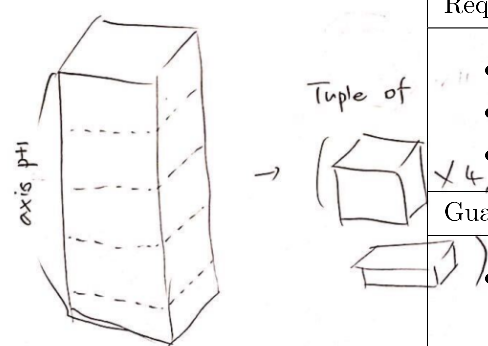
$$\frac{c = \{(d \neq 0) \land ((e-s)/d > 0)\}}{\sigma \vdash \mathtt{range}(s, e, d) \Rightarrow (1 + \lfloor (e-s)/d \rfloor), c} \qquad \text{Default: } s = 0, d = 1$$

`torch.Tensor.item`

<table>
<tr><td colspan="2"><code>a.item()</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• $|a| = ()$ or $|a| = (1, 1, 1, \ldots, 1)$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = e_n$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• Singular element tensor의 원소(스칼라 타입으로 반환)</td></tr>
</table>

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ c' = \{(\forall i = 1, 2, \ldots, k,\ e[i] = 1)\}\end{array}}{\sigma \vdash E.\texttt{item}() \Rightarrow e_n, c \cup c'}$$

`torch.split`

<table>
<tr><td colspan="2"><code>torch.split(x, n, p=0)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, \ldots, d_k)$
- $k \geq 1$
- $0 \leq p < k$

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

- 아래 proof tree와 같이 p+1번째 axis가 최대 n개의 원소를 가지도록 $\lceil d_{p+1}/n \rceil$개 튜플 형태로 반환

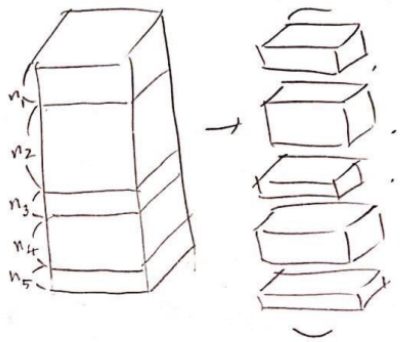
</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- Concat의 반대 역할 함수
- Divisible 여부 assert하지 않음
- 학습 및 테스트 데이터를 배치단위로 쪼개는 용도로 쓰일 것으로 추측

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

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = e[1{:}p]@(n)@e[p+2{:}k] \\ e_2 = e[1{:}p]@(n)@e[p+2{:}k] \\ \quad \ldots \\ e_{l-1} = e[1{:}p]@(n)@e[p+2{:}k] \\ e_l = e[1{:}p]@(n')@e[p+2{:}k] \quad \text{where } e[p+1] = n(l-1) + n',\ 0 < n' \leq n \\ c' = \{(k \geq 1) \wedge (0 \leq p < k)\}\end{array}}{\sigma \vdash \texttt{split}(E, n, p = 0) \Rightarrow (e_1, e_2, \ldots, e_l), c \cup c'}$$

l-원소 tuple 형태로 반환

<table>
<tr><td colspan="2"><code>torch.split(x, [n1, n2, ..., nl], p=0)</code></td></tr>
<tr><td rowspan="4">
</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, \dots, d_k)$
- $k \geq 1$
- $0 \leq p < k$
- $d_{p+1} = n_1 + n_2 + \cdots + n_l$

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

- 아래 proof tree와 같이 $p+1$번째 axis의 크기가 $n_1, n_2, \dots, n_l$인 l개의 텐서 튜플을 반환
- n_i의 합과 d_{p+1}이 같은지 assert

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

$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = e[1{:}p]@(n_1)@e[p+2{:}k] \\ e_2 = e[1{:}p]@(n_2)@e[p+2{:}k] \\ \quad \dots \\ e_l = e[1{:}p]@(n_l)@e[p+2{:}k] \\ c' = \{(k \geq 1) \land (0 \leq x < k) \land (e[p+1] = n_1 + n_2 + \cdots + n_l)\} \end{array}}{\sigma \vdash \texttt{split}(E, [n_1, n_2, \dots, n_l], p = 0) \Rightarrow (e_1, e_2, \dots, e_l), c \cup c'} \; l\text{-원소 tuple 형태로 반환}$$

`torch.zeros`, `torch.rand`, `torch.randn`

<table>
<tr><td colspan="2"><code>torch.zeros(t1, t2, ..., tl) or .rand, .randn</code></td></tr>
<tr><td rowspan="6"></td><td>Require</td></tr>
<tr><td></td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- $|y| = (t_1, t_2, \dots, t_l)$

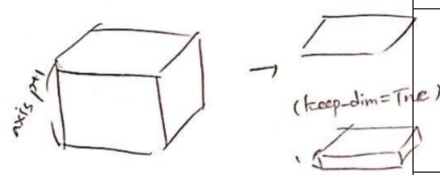
</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- 입력받은 형태대로 0, uniformly random, gaussian random 텐서를 반환

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

$$\forall \texttt{ft} \in \{\texttt{zeros}, \texttt{rand}, \texttt{randn}\}, \quad \frac{}{\sigma \vdash \texttt{ft}(t_1, t_2, \dots, t_l) \Rightarrow (t_1, t_2, \dots, t_l), \emptyset}$$

`torch.mode`

<table>
<tr><td colspan="2"><code>torch.mode(x, n=0, keep_dim=False)</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, \ldots, d_k)$
- $k \geq 1$
- $0 \leq n < k$

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

- $|y| = |z| = (d_1, d_2, \ldots, d_n, d_{n+2}, \ldots, d_k)$인 (y, z) 튜플 반환
- 세 번째 인자 $keep_dim$이 $True$이면 $|y| = |z| = (d_1, d_2, \ldots, d_n, 1, d_{n+2}, \ldots, d_k)$

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

- 입력받은 축을 기준으로한 통계적 최빈값 계산 함수

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

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{mode}(E, n=0) \Rightarrow (e', e'), c \cup c'}$$

tuple 형태로 반환

$$\frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{mode}(E, n=0, True) \Rightarrow (e', e'), c \cup c'}$$

tuple 형태로 반환

$$\frac{\sigma \vdash \texttt{mode}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \texttt{mode}(E, n=0, False) \Rightarrow (e, e), c}$$

tuple 형태로 반환

`torch.randint`

<table>
<tr><td colspan="2"><code>torch.randint(low=0, high, shape)</code></td></tr>
<tr><td rowspan="4"></td><td>Require</td></tr>
<tr><td>

- $low < high$
- $shape$이 well-defined인 텐서 shape. (스칼라 타입은 X)

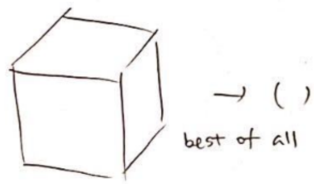
</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>

- $|y| = shape$

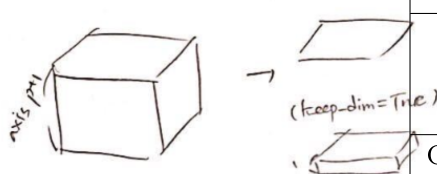
</td></tr>
</table>

$$\frac{low < high}{\sigma \vdash \texttt{randint}(low = 0, high, s) \Rightarrow tupleToShape(s), \emptyset}$$

torch.max, torch.min

`torch.max(x) or .min`	
	Require
	Guarantees
	• $\lvert y \rvert = ()$
	Comment
	• 주어진 텐서에서 최대/최소 값을 ()-shape 텐서로 반환(스칼라 X)

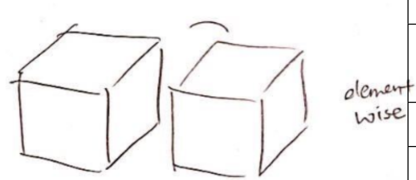
$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \texttt{ft}(E) \Rightarrow (), c}$$

`torch.max(x, n, keep_dim=False) or .min`	
	Require
	• $\lvert x \rvert = (d_1, d_2, \ldots, d_k)$ • $k \geq 1$, $0 \leq n < k$
	Guarantees
	• $\lvert y \rvert = \lvert z \rvert = (d_1, d_2, \ldots, d_n, d_{n+2}, \ldots, d_k)$인 (y, z) 튜플 반환 • 세 번째 인자 $keep_dim$이 $True$이면 $\lvert y \rvert = \lvert z \rvert = (d_1, d_2, \ldots, d_n, 1, d_{n+2}, \ldots, d_k)$
	Comment
	• 주어진 텐서에서 축 상의 최대/최소값과, 그에 해당하는 인덱스 번호를 쌍으로 엮어 튜플로 반환

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n] @ e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, n) \Rightarrow (e', e'), c \cup c'} \qquad \text{tuple 형태로 반환}$$

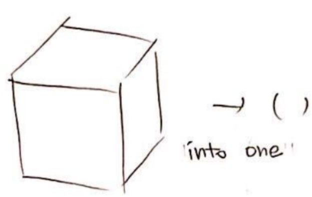
$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, n, True) \Rightarrow (e', e'), c \cup c'} \qquad \text{tuple 형태로 반환}$$

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\sigma \vdash \texttt{ft}(E, n) \Rightarrow (e, e), c}{\sigma \vdash \texttt{ft}(E, n, False) \Rightarrow (e, e), c} \qquad \text{tuple 형태로 반환}$$

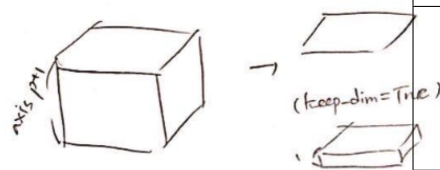
<table>
<tr><td colspan="2"><code>torch.max(x, y) or .min</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• $broadcastable(|x|, |y|)$</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $broadcast(|x|, |y|)$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 두 텐서의 elementwise 최대/최소</td></tr>
</table>

$$\forall \texttt{ft} \in \{\texttt{min}, \texttt{max}\}, \quad \frac{\begin{array}{c} \sigma \vdash E_1 \Rightarrow e_1, c_1 \\ \sigma \vdash E_2 \Rightarrow e_2, c_2 \end{array}}{\sigma \vdash \texttt{ft}(E_1, E_2) \Rightarrow broadcast(e_1, e_2), c_1 \cup c_2 \cup broadcastable(e_1, e_2)}$$

`torch.sum`, `torch.mean`

<table>
<tr><td colspan="2"><code>torch.sum(x) or .mean</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• <code>.mean</code>에 대해서는 텐서 타입이 floating이어야 함</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $|y| = ()$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 주어진 텐서의 합계/평균값을 ()-shape 텐서로 반환(스칼라 X)</td></tr>
</table>

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\sigma \vdash E \Rightarrow \_, c}{\sigma \vdash \texttt{ft}(E) \Rightarrow (), c}$$

<table>
<tr><td colspan="2"><code>torch.sum(x, n, keep_dim=False) or .mean</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, \ldots, d_k)$
- $k \geq 1$, $0 \leq n < k$
- `.mean`에 대해서는 텐서 타입이 floating이어야 함

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

- $|y| = (d_1, d_2, \ldots, d_n, d_{n+2}, \ldots, d_k)$
- 세 번째 인자 *keep_dim*이 $True$이면 $|y| = (d_1, d_2, \ldots, d_n, 1, d_{n+2}, \ldots, d_k)$

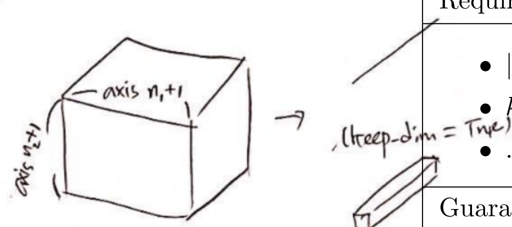
</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- 주어진 텐서에서 축 상의 합/평균을 반환

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

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{ft}(E, n) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l}\sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e' = e[1{:}n]@(1)@e[n+2{:}k] \\ c' = \{(k \geq 1) \wedge (0 \leq n < k)\}\end{array}}{\sigma \vdash \texttt{ft}(E, n, True) \Rightarrow e', c \cup c'}$$

<table>
<tr><td colspan="2"><code>torch.sum(x, [n1, n2, ..., nl], keep_dim=False) or .mean</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>

- $|x| = (d_1, d_2, \ldots, d_k)$
- $k \geq 1$, $0 \leq n < k$
- `.mean`에 대해서는 텐서 타입이 floating이어야 함

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

- $|y| = (d_{i_1}, d_{i_2}, \ldots, d_{i_{k-l}})$
  - $1, 2, \ldots, k$에서 $n_1 + 1, n_2 + 1, \ldots, n_l + 1$번째 항이 지워진 shape
- 세 번째 인자 *keep_dim*이 $True$이면 $n_1 + 1, n_2 + 1, \ldots, n_l + 1$번째 항은 삭제되지 않고 1로 남음

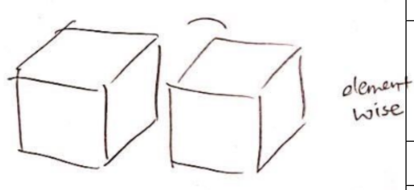
</td></tr>
<tr><td>Comment</td></tr>
<tr><td>

- 주어진 텐서에서 여러 축을 통합한 합/평균을 반환

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

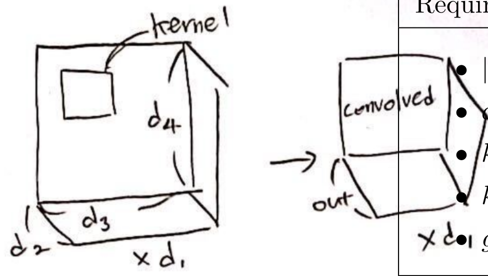
$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = \texttt{if } 0 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[1]) \\ e_2 = \texttt{if } 1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[2]) \\ \quad \cdots \\ e_k = \texttt{if } k-1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } () \texttt{ else } (e[k]) \\ e' = e_1 @ e_2 @ \cdots @ e_k \\ c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \ldots, r,\ 0 \leq n_i < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, (n_1, n_2, \ldots, n_r)) \Rightarrow e', c \cup c'}$$

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ e_1 = \texttt{if } 0 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[1]) \\ e_2 = \texttt{if } 1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[2]) \\ \quad \cdots \\ e_k = \texttt{if } k-1 \in \{n_1, n_2, \ldots, n_r\} \texttt{ then } (1) \texttt{ else } (e[k]) \\ e' = e_1 @ e_2 @ \cdots @ e_k \\ c' = \{(k \geq 1) \wedge (\forall i = 1, 2, \ldots, r,\ 0 \leq n_i < k)\} \end{array}}{\sigma \vdash \texttt{ft}(E, (n_1, n_2, \ldots, n_r), True) \Rightarrow e', c \cup c'}$$

<table>
<tr><td colspan="2"><code>torch.sum(x, y) or .mean</code></td></tr>
<tr><td rowspan="6">
</td><td>Require</td></tr>
<tr><td>• $broadcastable(|x|, |y|)$
• .mean에 대해서는 텐서 타입이 floating이어야 함</td></tr>
<tr><td>Guarantees</td></tr>
<tr><td>• $broadcast(|x|, |y|)$</td></tr>
<tr><td>Comment</td></tr>
<tr><td>• 두 텐서의 elementwise 합/평균</td></tr>
</table>

$$\forall \texttt{ft} \in \{\texttt{sum}, \texttt{mean}\}, \quad \frac{\sigma \vdash \texttt{ft}(E, X) \Rightarrow e, c}{\sigma \vdash \texttt{ft}(E, X, False) \Rightarrow e, c}$$

`torch.nn.Conv2d`

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

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

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

- (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.

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

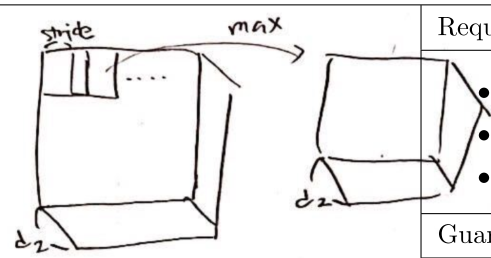
$$\frac{\begin{array}{l} \sigma \vdash E \Rightarrow e, c \\ k = \texttt{rank}(e) \\ h = \left\lfloor \frac{e[3] + 2 \times padding[0] - dilation[0] \times (kernel\_size[0] - 1) - 1}{stride[0]} \right\rfloor + 1 \\ w = \left\lfloor \frac{e[4] + 2 \times padding[1] - dilation[1] \times (kernel\_size[1]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

$kernel_size, stride, padding, dilation$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

`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)
- $kernel\_size[0] \le d_3 + 2 \times padding[0]$
- $kernel\_size[1] \le d_4 + 2 \times padding[1]$

</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>

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

$kernel_size, stride, padding, dilation$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>

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

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

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