# Matmul Layers

`torch.nn.Linear`

| torch.nn.Linear(in_features, out_features, bias=True)(x) | |
|---|---|
| | Require |
| | • $\lvert x \rvert = (d_1, d_2, \ldots, d_k)$ <br> • $\texttt{rank}(\lvert x \rvert) \geq 1$ <br> • $d_k = in\_features$ |
| | Guarantees |
| | • $\lvert y \rvert = (d_1, d_2, \ldots, d_{k-1}, out\_features)$ |
| | Comment |
| | • $y = xA^T + b$를 계산하는 dense 레이어 <br> • 1차원인 경우에도 잘 작동합니다. <br> • $bias$ 옵션은 출력 shape에 영향을 주지 않습니다. |

$$\sigma \vdash E \Rightarrow e, c$$
$$e' = e[1 : k - 1]@(out)$$
$$c' = \{(\texttt{rank}(e) \geq 1) \wedge (d_k = in)\}$$
$$\overline{\sigma \vdash \texttt{Linear}(in, out, bias = True)(E) \Rightarrow e', c \cup c'}$$

# Activations

`torch.nn.ReLU, torch.nn.ReLU6, torch.relu, torch.nn.functional.relu`

| torch.nn.ReLU(inplace=True)(x) | |
|---|---|
| | Require |
| | |
| | Guarantees |
| | • $\lvert y \rvert = \lvert x \rvert$ (same shape) |
| | Comment |
| | • $inplace$ 옵션은 shape에 영향을 주지 않습니다. <br> • ReLU6도 ReLU와 똑같은 방식으로 shape 계산 <br> • Bulitins인 `torch.relu`와 `torch.nn.functional.relu`는 같은거 |

$$\forall \texttt{ft} \in \{\texttt{ReLU}, \texttt{ReLU6}\}, \qquad \frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \texttt{ft}(inplace = True)(E) \Rightarrow e, c}$$

$$\frac{\sigma \vdash E \Rightarrow e, c}{\sigma \vdash \texttt{relu}(E, inplace = True) \Rightarrow e, c}$$

# End Points

`torch.nn.CrossEntropyLoss`

| `torch.nn.CrossEntropyLoss(weight=None, size_average=None, ignore_index=-100, reduction='mean')(input, target)` | |
|---|---|
| | **Require** |
| | • $\|input\| = (n, c, d_1, d_2, \ldots, d_k)$ $\quad$ ($\texttt{rank}(\|input\|) \geq 2$) <br> • $\|target\| = (n, d_1, d_2, \ldots, d_k)$ <br> $\quad$ – $\texttt{rank}(\|target\|) + 1 = \texttt{rank}(\|input\|)$ <br> $\quad$ – $\|target\|[1] = \|input\|[1]$, $\|target\|[2] = \|input\|[3]$, ... <br> • if $weight \neq None$, then $\|weight\| = (c)$ |
| | **Guarantees** |
| | • $\|y\| = \texttt{if } reduction == \text{`none'} \texttt{ then } (n, d_1, d_2, \ldots, d_k) \texttt{ else } ()$ |
| | **Comment** |
| | • $size\_average$, $reduce$ 인자는 deprecated로 도큐먼트되어 있습니다. <br> • $ignore\_index$는 $reduction$이나 $size\_average$ 옵션에서 평균을 계산하면서 생략하는 인덱스 번호를 지정할 때 쓰는 것으로, shape에 영향을 주지 않습니다. |

$$\sigma \vdash E \Rightarrow e, c_e$$
$$\sigma \vdash T \Rightarrow t, c_t$$
$$\sigma \vdash weight \Rightarrow w, c_w \quad \text{if } weight \neq None, \text{ otherwise } c_w = \emptyset$$
$$c_{dim} = \{(\texttt{rank}(e) \geq 2) \wedge (\texttt{rank}(e) = \texttt{rank}(t) + 1)\}$$
$$c_{elt} = \{(e[1] = t[1]) \wedge (e[3] = t[2]) \wedge (e[4] = t[3]) \wedge \cdots\}$$
$$c_{weight} = \{(weight = None \vee w = (e[2]))\}$$
$$e' = \texttt{if } reduction = \text{`none'} \texttt{ then } t \texttt{ else } ()$$

$$\overline{\sigma \vdash \texttt{CrossEntropyLoss}(weight = None, ..., reduction = \text{`mean'})(E, T) \Rightarrow e', c_e \cup c_t \cup c_w \cup c_{dim} \cup c_{elt} \cup c_{weight}}$$

`torch.nn.TripletMarginLoss`

**NOT DONE YET!!**

Reading Papers...

| torch.nn.TripletMarginLoss(margin=1.0, p=2.0, eps=1e-6, swap=False, size_average=False, reduce=None, reduction='mean')(anchor, positive, negative) | |
|---|---|
| | **Require** |
| | • $broadcastable(\|anchor\|, \|positive\|, \|negative\|)$<br>• $\texttt{rank}(broadcast(\|anchor\|, \|positive\|, \|negative\|)) \geq 2$<br>• if $swap$ is $True$ then,<br>  – $\texttt{rank}(\|anchor\|), \|positive\|, \|negative\|)$<br>  – $\texttt{rank}(broadcast(\|anchor\|, \|positive\|, \|negative\|)) \geq 2$<br>• $\|target\| = (n, d_1, d_2, \ldots, d_k)$<br>  – $\texttt{rank}(\|target\|) + 1 = \texttt{rank}(\|input\|)$<br>  – $\|target\|[1] = \|input\|[1], \|target\|[2] = \|input\|[3], \ldots$<br>• if $weight \neq None$, then $\|weight\| = (c)$ |
| | **Guarantees** |
| | • $\|y\| = \texttt{if}\ reduction == \text{`none'}\ \texttt{then}\ (n, d_1, d_2, \ldots, d_k)\ \texttt{else}\ ()$ |
| | **Comment** |
| | • $size\_average$, $reduce$ 인자는 deprecated로 도큐먼트되어 있습니다.<br>• $ignore\_index$는 $reduction$이나 $size\_average$ 옵션에서 평균을 계산하면서 생략하는 인덱스 번호를 지정할 때 쓰는 것으로, shape에 영향을 주지 않습니다.<br>• 토치 구현의 버그인지 모르겠는데, $swap$이 $True$인 경우가 너무 복잡합니다.. 도큐먼트에도 논문 하나만 첨부되어 있습니다. |

$$\sigma \vdash E \Rightarrow e, c_e$$
$$\sigma \vdash T \Rightarrow t, c_t$$
$$\sigma \vdash weight \Rightarrow w, c_w \quad \text{if } weight \neq None, \text{ otherwise } c_w = \emptyset$$
$$c_{dim} = \{(\texttt{rank}(e) \geq 2) \wedge (\texttt{rank}(e) = \texttt{rank}(t) + 1)\}$$
$$c_{elt} = \{(e[1] = t[1]) \wedge (e[3] = t[2]) \wedge (e[4] = t[3]) \wedge \cdots\}$$
$$c_{weight} = \{(weight = None \vee w = (e[2]))\}$$
$$e' = \texttt{if } reduction = \text{`none'}\ \texttt{then}\ t\ \texttt{else}\ ()$$

$$\overline{\sigma \vdash \texttt{CrossEntropyLoss}(weight = None, ..., reduction = \text{`mean'})(E, T) \Rightarrow e', c_e \cup c_t \cup c_w \cup c_{dim} \cup c_{elt} \cup c_{weight}}$$

## Technique

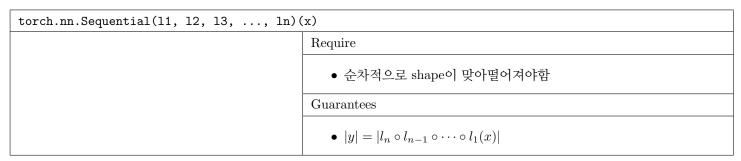`torch.nn.Dropout, torch.dropout, torch.nn.functional.dropout`

| torch.nn.Dropout(...)(x) | |
|---|---|
| | **Require** |
| | |
| | **Guarantees** |
| | • $\|y\| = \|x\|$ (same shape) |
| | **Comment** |
| | • 모든 옵션은 shape에 영향을 주지 않습니다.<br>• Bulitins인 `torch.dropout`와 `torch.nn.functional.dropout`는 서로 역할이 같습니다. |

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

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

# Wrapper

torch.nn.Sequential

| torch.nn.Sequential(l1, l2, l3, ..., ln)(x) | |
|---|---|
| | Require |
| | • 순차적으로 shape이 맞아떨어져야함 |
| | Guarantees |
| | • $\lvert y \rvert = \lvert l_n \circ l_{n-1} \circ \cdots \circ l_1(x) \rvert$ |

$$\frac{\sigma \vdash l_n \circ l_{n-1} \circ \cdots l_1(E) \Rightarrow e, c}{\sigma \vdash \texttt{Sequential}(l_1, l_2, \ldots, l_n)(E) \Rightarrow e, c}$$