<QR-DQN: How I coded.>

※ The first dim of tensors should be mb_size, but we are ignoring it.

· target_q_dist = $\begin{pmatrix} target_1 & \cdots & target_N \\ \vdots & \ddots & \vdots \\ target_1 & \cdots & target_N \end{pmatrix}$ ($N$ : # of quantiles) = n_quant.

eval_q_dist = $\begin{pmatrix} eval_1 & \cdots & eval_1 \\ \vdots & \ddots & \vdots \\ eval_N & \cdots & eval_N \end{pmatrix}$

If we broadcast it,
target_q_dist
$\in \mathbb{R}^{mb \times 1 \times N\_quant\,(target)}$
$\rightarrow \mathbb{R}^{mb \times N\_quant\,(eval) \times N\_quant\,(target)}$
eval_q_dist expands as well!

u_values = $\begin{pmatrix} target_1 - eval_1 & \cdots & target_N - eval_1 \\ \vdots & \ddots & \vdots \\ target_1 - eval_N & \cdots & target_N - eval_N \end{pmatrix} \in \mathbb{R}^{mb \times N\_quant\,(eval) \times N\_quant\,(target)}$

tau_values = $\begin{pmatrix} \tau_1 & \cdots & \tau_1 \\ \vdots & \ddots & \vdots \\ \tau_N & \cdots & \tau_N \end{pmatrix} \in \mathbb{R}^{mb \times N\_quant\,(eval) \times N\_quant\,(target)}$

u_values.le(0) = $\begin{pmatrix} 1(target_1 - eval_1 < 0) & \cdots & 1(target_N - eval_1 < 0) \\ \vdots & \ddots & \vdots \\ 1(target_1 - eval_N < 0) & \cdots & 1(target_N - eval_N < 0) \end{pmatrix}$

$\in \mathbb{R}^{mb \times N\_quant\,(eval) \times N\_quant\,(target)}$

weight = $\begin{pmatrix} |\tau_1 - 1(target_1 - eval_1 < 0)| & \cdots & |\tau_1 - 1(target_N - eval_1 < 0)| \\ \vdots & \ddots & \vdots \\ |\tau_N - 1(target_1 - eval_N < 0)| & \cdots & |\tau_N - 1(target_N - eval_N < 0)| \end{pmatrix}$

$\in \mathbb{R}^{mb \times N\_quant\,(eval) \times N\_quant\,(target)}$

Although N_quant (eval) = N_quant (target), we need to differentiate which dimension corresponds to which (eval/target).

$$\text{rho\_values} = \begin{pmatrix} L_K(\text{eval}_1 - \text{target}_1) & \cdots & L_K(\text{eval}_1 - \text{target}_N) \\ \vdots & \ddots & \vdots \\ L_K(\text{eval}_N - \text{target}_1) & \cdots & L_K(\text{eval}_N - \text{target}_N) \end{pmatrix}$$

$$\in \mathbb{R}^{mb \times N\_quant(eval) \times N\_quant(target)}$$

weight * rho_values

$$= \begin{pmatrix} |\tau_1 - 1(\text{target}_1 - \text{eval}_1 < 0)| \cdot L_K(\text{eval}_1 - \text{target}_1) & \cdots & |\tau_1 - 1(\text{target}_N - \text{eval}_1 < 0)| \cdot L_K(\text{eval}_1 - \text{target}_N) \\ \vdots & \ddots & \vdots \\ |\tau_N - 1(\text{target}_1 - \text{eval}_N < 0)| \cdot L_K(\text{eval}_N - \text{target}_1) & \cdots & |\tau_N - 1(\text{target}_N - \text{eval}_N < 0)| \cdot L_K(\text{eval}_N - \text{target}_N) \end{pmatrix}$$

$$\in \mathbb{R}^{mb \times N\_quant(eval) \times N\_quant(target)}$$

- Now we can compute our objective function

$$\sum_{i=1}^{N} E_j \left[ \rho^{K}_{\hat{\tau}_i} (TQ_j - Q_i)(x,a) \right]$$

eval_index    target_index

(weight * rho_values)    necessary!

$$= \sum_{i=1}^{N} E_j \left[ |\hat{\tau}_i - 1\{(TQ_j - Q_i)(x,a) < 0\}| \cdot L_K\{(TQ_j - Q_i)(x,a)\} / K \right].$$

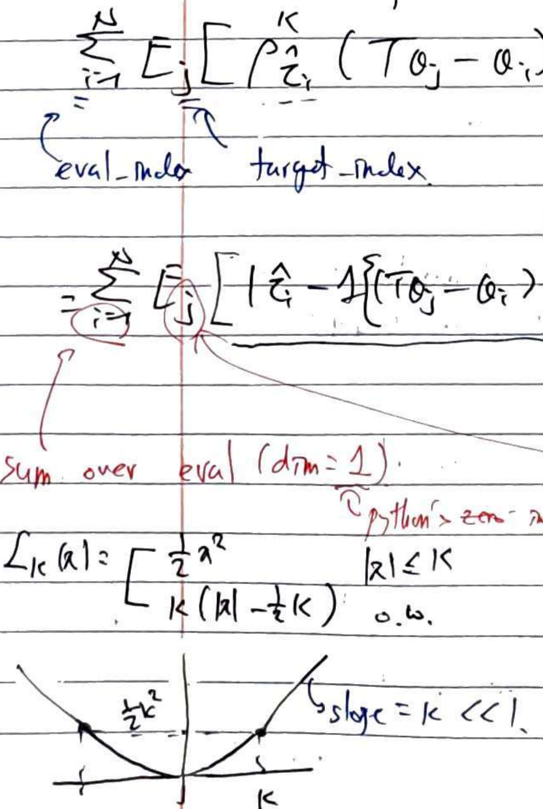
depends on sample (m=1, ..., mb_size)

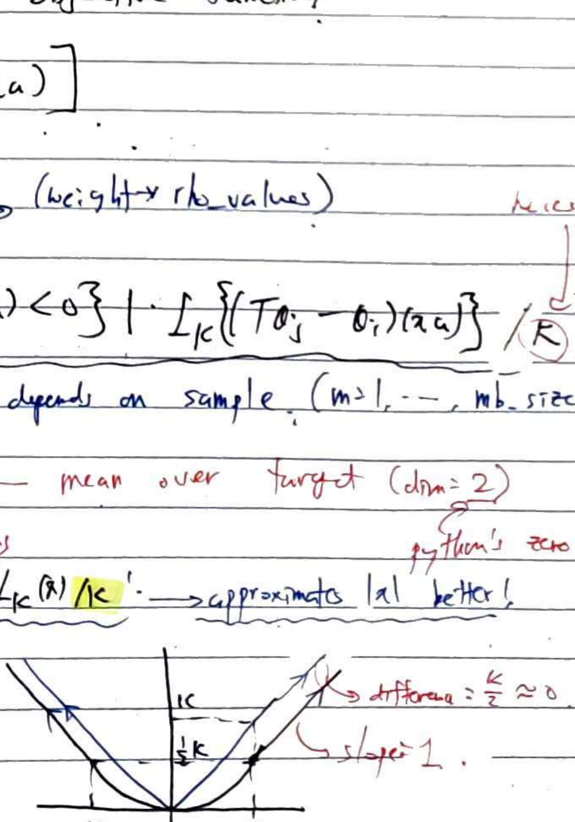
sum over eval (dim=1)    mean over target (dim=2)

python's zero-indexing    python's zero-indexing

- $L_K(x) = \begin{cases} \frac{1}{2}x^2 & |x| \le K \\ K(|x| - \frac{1}{2}K) & \text{o.w.} \end{cases}$

$L_K(x)/K$ → approximates $|x|$ better!



"smoothly" (differentiable)

↳ Since the goal of Huber loss is to approximate $|x|$, which is the objective in Quantile-regression, it makes more sense to use $\mathcal{L}_\kappa(x)/\kappa$ instead of $\mathcal{L}_\kappa(x)$.

## <IQN: understanding the mechanism>

ignoring the first dimension (mb_size)

- Let $\psi: \mathbb{R}^{4\times 84\times 84} \longrightarrow \mathbb{R}^{(64\times n\times n)}$ : 3 repetitive pairs of (CNN, ReLU),
  (3d-tensor) (1d-tensor) followed by Flatten → (1d-tensor)

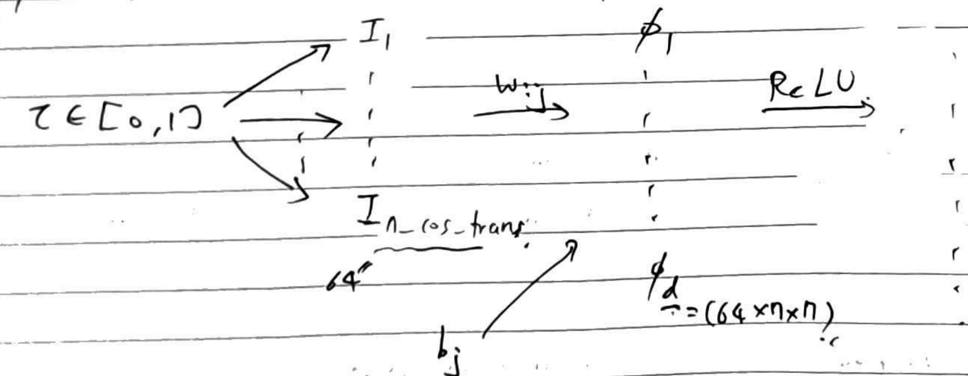
  $f: \mathbb{R}^{(64\times n\times n)} \longrightarrow \mathbb{R}^{|A|}$ : 2 fully-connected layers each being $f_1: \mathbb{R}^{(64\times n\times n)} \to \mathbb{R}^{512}$, $f_2: \mathbb{R}^{512} \to \mathbb{R}^{|A|}$.
  $(f = f_2 \circ f_1)$

  $\phi: [0,1] \longrightarrow \mathbb{R}^{(64\times n\times n)}$ : to be explained later on.

- For a fixed value of $\tau_0 \in [0,1]$,

$$Z_{\tau_0}(x,a) = f(\psi(x)\odot\phi(\tau_0)) \quad (\in \mathbb{R}^{|A|})$$
$$= (\underbrace{f_1(\psi(x)\odot\phi(\tau_0))}_{\in\mathbb{R}}, \ldots, \underbrace{f_{|A|}(\psi(x)\odot\phi(\tau_0))}_{\in\mathbb{R}})^T \quad (\in \mathbb{R}^{|A|})$$

- $\phi: [0,1] \to \mathbb{R}^{(64\times n\times n)}$ can be explained as follows,




where $I_i = \cos(\pi \cdot i \cdot \tau)$.

$(w_{ij})_{i=1,\ldots,n\_cos\_trans;\ j=1,\ldots,d}$

$(b_j)_{j=1,\ldots,d}$

· Then our final NN can be summarized as

$F : \mathbb{R}^{4\times 84\times 84} \times [0,1] \to \mathbb{R}^{|A|}$, which is defined as follows

there will be mb_size samples of images

there will be N_quant (either eval or target) samples.

for online network / for target network

$$F(x;\tau) = Z_\tau(x) = \big(\underbrace{Z_\tau(x,a_1)}_{\tau\text{-th quantile of } Z(x,a_1)} \cdots Z_\tau(x,a_{|A|})\big)^T \; (\in \mathbb{R}^{|A|})$$

$$= \big(f_1(\psi(x)\odot\phi(\tau)), \;\cdots,\; f_{|A|}(\psi(x)\odot\phi(\tau))\big)^T \; (\in \mathbb{R}^{|A|})$$

· Note that all the fcts $f = (f_1 \cdots f_{|A|})^T$, $\psi$, $\phi$ are NN's that we should train.

Since there are "mb_size" many samples of $x_i$, and "N_quant" (eval or target) many samples of $\tau_i$, there are (mb_size × N_quant) many samples that we need (or (mb_size × N_quant (eval) × N_quant (target)) to be mindful of.

· cos_trans

$$\begin{array}{c c} & \begin{array}{ccc} i=0 & \cdots & i = n\_cos\_trans-1 \end{array} \\ \begin{array}{c} \tau_1 \\ \vdots \\ \tau_{N\_quant} \end{array} & \begin{pmatrix} \cos(\pi\cdot 0\cdot \tau_1) & \cdots & \cos(\pi\cdot(n-1)\cdot\tau_1) \\ \vdots & \ddots & \\ \cos(\pi\cdot 0\cdot \tau_{N\_quant}) & & \cos(\pi\cdot(n-1)\cdot\tau_{N\_quant}) \end{pmatrix} \in \mathbb{R}^{N\_quant \times n\_cos\_trans} \end{array}$$

treated as # of samples in network within $\phi : [0,1] \to \mathbb{R}^{n\_cos\_trans} \to \mathbb{R}^{512}$

deterministic fct that we don't learn.

network part