We made an important finding when re-considering the default settings we set for the various parameters of our operators: $m, b, y \in \mathbb{R}_m$ (1.128), $\varepsilon, \ell$ (1.181), cost $h$ (1.207) and rescaling of input values (1.214). Although all of these choices converge to standard sort (regardless of $y, h$ or of the rescaling) as soon as $n = m$, $a = b$ and $\varepsilon \to 0$, these choices do impact the “shape” of the differentiability encoded in our operators when $\varepsilon > 0$ (see p.6). Our finding is that our initial choice to do a min-max or a softmax rescaling of the input arrays into $[0, 1]$ as in 1.214 was not a good choice, leading to instabilities and “squeezed” values. Applying instead a logistic map on standardised input values (in batchnorm fashion) leads to improved results across the board (see below): Vanilla CNNs trained on CIFAR-10/100 with the soft-error loss (1.248) beat on average those trained with XE; RESNETs yield comparative results with both losses; In regression, using the soft-quantile loss (Eq.4) yields SOTA results when minimizing median error ($\tau = 0.5$), but mixed for small $\tau = 0.1, 0.2$ (current understanding: soft-quantiles are computed on mini-batches, therefore our loss is differentiable but biased, while the pinball loss is not-differentiable but unbiased). Finally, we beat neuralsort [11] on the setup shared in their colab. We are now ready to share our code, for others to build upon.

Our new results are more convincing. [13] top row (uniform, presumably?) You are right.

**Reviewer #2:** I think the basic idea is to do stuff like this [...]. Your pseudo-code agrees with the spirit of our work. However, we argue that our implementation is far more effective, owing to the flexibility given by weights $b$ and number of targets $m$. Writing $n = \text{len}(Xs)$ and $\ell = \text{n\_iterations}$, your implementation requires $n^2 \cdot \ell$ operations. Ours has linear complexity $n \cdot m \cdot \ell$ (see 1.225). When computing a single quantile—the case you consider in your pseudo-code—we bring $m$ down to 3 (see 1.234, and bottom row of Fig.3 where $m = 5$). Finally, we integrate directly quant within $b$, no need to round to get an index (see Figs. 2 & 1(b)). I actually found the motivation distracting. As mentioned to R1, we have moved references to OT from §1 to §2.3. The rigorous derivation of our tools is now encapsulated in §2.3, which can be skipped on a first read. We will "sell" in §1 our operators as "no-brainer" plug-ins for sorting, and encapsulate all the OT discussion in §2.3. [...]

**Reviewer #3:** More convincing numerical experiments would certainly improve the paper. We are excited to report more convincing experiments. [...] considered certain algorithms that involve sorting as intermediate steps. Your point illustrates perfectly our motivation and we are genuinely excited by the opportunities given by this new tool. For instance, quantile-based losses are related to fairness (e.g. arxiv 1907.08646) and we are keen to investigate connections.