Supplementary Material for "Rank & Sort Loss for Object Detection and Instance Segmentation"

Kemal Oksuz, Baris Can Cam, Emre Akbas*, Sinan Kalkan*
Dept. of Computer Engineering, Middle East Technical University, Ankara, Turkey
{kemal.oksuz, can.cam, eakbas, skalkan}@metu.edu.tr

Contents

S1. Details of RS Loss

In this section, we present the derivations of gradients and obtain the loss value and gradients of RS Loss on an example in order to provide more insight.

S1.1. Derivation of the Gradients

The gradients of a ranking-based loss function can be determined as follows. Eq. 3 in the paper states that:

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z}\left(\sum_{j\in\mathcal{P}\cup\mathcal{N}} \Delta x_{ji} - \sum_{j\in\mathcal{P}\cup\mathcal{N}} \Delta x_{ij}\right). \tag{S1}$$

*Equal contribution for senior authorship.

Our identity update reformulation suggests replacing Δx_{ij} by L_{ij} which yields:

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z}\left(\sum_{j\in\mathcal{P}\cup\mathcal{N}} L_{ji} - \sum_{j\in\mathcal{P}\cup\mathcal{N}} L_{ij}\right). \tag{S2}$$

We split both summations into two based on the labels of the examples, and express $\frac{\partial \mathcal{L}}{\partial s_i}$ using four terms:

$$\frac{\partial \mathcal{L}}{\partial s_i} = \frac{1}{Z}\left(\sum_{j\in\mathcal{P}} L_{ji} + \sum_{j\in\mathcal{N}} L_{ji} - \sum_{j\in\mathcal{P}} L_{ij} - \sum_{j\in\mathcal{N}} L_{ij}\right). \tag{S3}$$

Then simply by using the primary terms of RS Loss, defined in Eq. 9 in the paper as:

$$L_{ij} = \begin{cases} \left(\ell_{\mathrm{R}}(i) - \ell^*_{\mathrm{R}}(i)\right) p_R(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{N} \\ \left(\ell_{\mathrm{S}}(i) - \ell^*_{\mathrm{S}}(i)\right) p_S(j|i), & \text{for } i \in \mathcal{P}, j \in \mathcal{P}, \\ 0, & \text{otherwise,} \end{cases} \tag{S4}$$

With the primary term definitions, we obtain the gradients of RS Loss using Eq. S3.

Gradients for $i \in \mathcal{N}$. For $i \in \mathcal{N}$, we can respectively express the four terms in Eq. S3 as follows:

- $\sum_{j\in\mathcal{P}} L_{ji} = \sum_{j\in\mathcal{P}} \left(\ell_{\mathrm{R}}(j) - \ell^*_{\mathrm{R}}(j)\right) p_R(i|j)$,
- $\sum_{j\in\mathcal{N}} L_{ji} = 0$ (no negative-to-negative error is defined for RS Loss – see Eq. S4),
- $\sum_{j\in\mathcal{P}} L_{ij} = 0$ (no error when $j \in \mathcal{P}$ and $i \in \mathcal{N}$ for L_{ij} – see Eq. S4),
- $\sum_{j\in\mathcal{N}} L_{ij} = 0$ (no negative-to-negative error is defined for RS Loss – see Eq. S4),

which, then, can be expressed as (by also replacing $Z = |\mathcal{P}|$ following the definition of RS Loss):

$$\frac{\partial \mathcal{L}_{\text{RS}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\left(\sum_{j\in\mathcal{P}} L_{ji} + \cancelto{0}{\sum_{j\in\mathcal{N}} L_{ji}} - \cancelto{0}{\sum_{j\in\mathcal{P}} L_{ij}} - \cancelto{0}{\sum_{j\in\mathcal{N}} L_{ij}}\right), \tag{S5}$$

$$= \frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}}\left(\ell_{\text{R}}(j) - \cancelto{0}{\ell^*_{\text{R}}(j)}\right) p_R(i|j), \tag{S6}$$

$$= \frac{1}{|\mathcal{P}|}\sum_{j\in\mathcal{P}} \ell_{\text{R}}(j) p_R(i|j), \tag{S7}$$

concluding the derivation of the gradients if $i \in \mathcal{N}$.

Gradients for $i \in \mathcal{P}$. We follow the same methodology for $i \in \mathcal{P}$ and express the same four terms as follows:

- $\sum_{j\in\mathcal{P}} L_{ji} = \sum_{j\in\mathcal{P}} \left(\ell_{\text{S}}(j) - \ell^*_{\text{S}}(j)\right) p_S(i|j)$,
- $\sum_{j\in\mathcal{N}} L_{ji} = 0$ (no error when $j \in \mathcal{N}$ and $i \in \mathcal{P}$ for L_{ji} – see Eq. S4),
- $\sum_{j\in\mathcal{P}} L_{ij}$ reduces to $\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)$ simply by rearranging the terms and $\sum_{j\in\mathcal{P}} p_S(j|i) = 1$ since $p_S(j|i)$ is a pmf:

$$\sum_{j\in\mathcal{P}} L_{ij} = \sum_{j\in\mathcal{P}} \left(\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)\right) p_S(j|i), \tag{S8}$$

$$= \left(\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)\right) \cancelto{1}{\sum_{j\in\mathcal{P}} p_S(j|i)}, \tag{S9}$$

$$= \ell_{\text{S}}(i) - \ell^*_{\text{S}}(i). \tag{S10}$$

- Similarly, $\sum_{j\in\mathcal{N}} L_{ij}$ reduces to $\ell_{\text{R}}(i) - \ell^*_{\text{R}}(i)$:

$$\sum_{j\in\mathcal{N}} L_{ij} = \sum_{j\in\mathcal{N}} \left(\ell_{\text{R}}(i) - \ell^*_{\text{R}}(i)\right) p_R(j|i) \tag{S11}$$

$$= \left(\ell_{\text{R}}(i) - \ell^*_{\text{R}}(i)\right) \cancelto{1}{\sum_{j\in\mathcal{N}} p_R(j|i)} \tag{S12}$$

$$= \ell_{\text{R}}(i) - \ell^*_{\text{R}}(i). \tag{S13}$$

Combining these four cases together, we have the following gradient for $i \in \mathcal{P}$:

$$\frac{\partial \mathcal{L}_{\text{RS}}}{\partial s_i} = \frac{1}{|\mathcal{P}|}\left(\sum_{j\in\mathcal{P}} \left(\ell_{\text{S}}(j) - \ell^*_{\text{S}}(j)\right) p_S(i|j) \right. \tag{S14}$$

$$\left. - \left(\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)\right) - \left(\ell_{\text{R}}(i) - \ell^*_{\text{R}}(i)\right)\right). \tag{S15}$$

Input id, i ($i \in \mathcal{P}$ is underlined)	0	1	2	3	4	5	6	7
(a) Input of RS Loss (positive if $y_i > 0$; else negative)								
Logits, s_i	3.00	2.00	1.00	0.00	-1.00	-2.00	-3.00	-4.00
Labels, y_i	0.90	0.40	0.00	0.00	0.80	0.00	0.10	0.00
(b) Current&target error and RS Loss on each $i \in \mathcal{P}$ (N/A: negatives; bold: non-zero loss)								
Current ranking error, $\ell_R(i)$	0.00	0.00	N/A	N/A	0.40	N/A	0.42	N/A
Current sorting error, $\ell_S(i)$	0.10	0.35	N/A	N/A	0.30	N/A	0.38	N/A
Target ranking error, $\ell^*_R(i)$	0.00	0.00	N/A	N/A	0.00	N/A	0.00	N/A
Target sorting error, $\ell^*_S(i)$	0.10	0.35	N/A	N/A	0.15	N/A	0.38	N/A
Ranking Loss, $\ell_R(i) - \ell^*_R(i)$	0.00	0.00	N/A	N/A	**0.40**	N/A	**0.42**	N/A
Sorting Loss, $\ell_S(i) - \ell^*_S(i)$	0.00	0.00	N/A	N/A	**0.15**	N/A	0.00	N/A
Total Loss, $(\ell_R(i)+\ell_S(i)) - (\ell^*_R(i) + \ell^*_S(i))$	0.00	0.00	N/A	N/A	**0.55**	N/A	**0.42**	N/A
RS Loss, $\mathcal{L}_{RS}$ (average over total losses) : 0.24								

Figure S1. An example illustrating the computation of RS Loss. (a) The inputs of RS Loss are logits and continuous labels (i.e. IoU). (b) Thanks to the "Identity Update" (Section 3.2 in the paper), the loss computed on each example considers its target error, hence, it is interpretable. E.g. $i = 0, 1, 6$ have positive current sorting errors, but already sorted properly among examples with larger logits, which can be misleading when loss value ignores the target error. Since RS Loss is computed only over positives, N/A is assigned for negatives.

Finally, for clarity, we rearrange the terms also by using $\ell^*_{\text{RS}}(i) - \ell_{\text{RS}}(i) = -(\ell_{\text{S}}(i) - \ell^*_{\text{S}}(i)) - (\ell_{\text{R}}(i) - \ell^*_{\text{R}}(i))$:

$$\frac{1}{|\mathcal{P}|}\left(\ell^*_{\text{RS}}(i) - \ell_{\text{RS}}(i) + \sum_{j\in\mathcal{P}} \left(\ell_{\text{S}}(j) - \ell^*_{\text{S}}(j)\right) p_S(i|j)\right), \tag{S16}$$

concluding the derivation of the gradients when $i \in \mathcal{P}$.

S1.2. Insights on the Computation of RS Loss and its Gradients on an Example

In Fig. S1, we illustrate the input and the computation of RS Loss. We emphasize that our Identity Update provides interpretable loss values when the target value is non-zero (Fig. S1(b)). Previous work [2, 13] fail to provide interpretable loss values.

Computation of the Loss. To compute the loss following the three-step algorithm in the paper, the first and the third steps are trivial (Fig. 2 in the paper), thus, here we present in Fig. S2 (a) how the primary terms (L_{ij}) are computed for our example (Fig. S1) following Eq. S4.

Optimization of the Loss. Fig. S2(b) presents and discusses the gradients obtained using Eq. S5, S16.

S2. Analyses

Section S2.1 presents our experiments to validate our design choices and Section S2.2 discusses the drawbacks of aLRP Loss, and how we fix them.

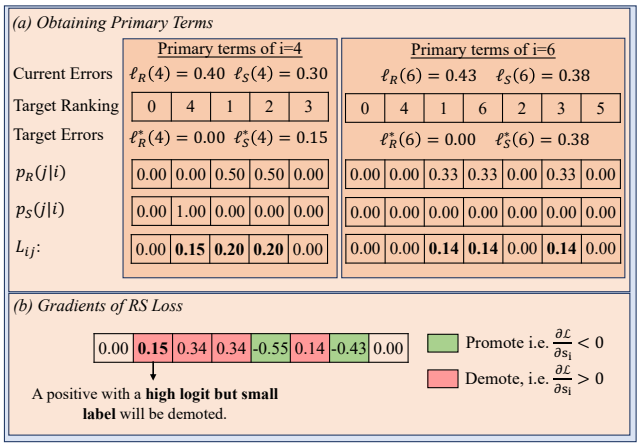

Figure S2. An example illustrating the computation of primary terms in RS Loss. (a) The computation of primary terms. We only show the computation for positives $i = 4$ and $i = 6$ since for $i = 0$ and $i = 1$ the total loss is 0 (see Fig. S1(b)); and RS Loss does not compute error on negatives by definition (i.e. discretizes the space only on positives). To compute primary terms, L_{ij}, one needs current errors, target errors and pmfs for both ranking and sorting, which are included in the figure respectively. In order to compute the target errors on a positive $i \in \mathcal{P}$, the examples are first thresholded from s_i and the ones with larger (i.e. $s_j \geq s_i$) logits are obtained. Then, target rankings are identified using continuous labels. The ranking and sorting errors computed for the target ranking determines target errors, $\ell^*_R(i)$ and $\ell^*_S(i)$. The ranking and sorting losses, $\ell_R(i) - \ell^*_R(i)$ and $\ell_S(i) - \ell^*_S(i)$ respectively, are then distributed over examples causing these losses uniformly via pmfs $p_S(j|i)$ and $p_R(j|i)$ to determine pairwise errors, i.e. primary terms. (b) The gradients are obtained simply by using primary terms as the update in Eq. S1 following identity update yielding Eq. S5 and S16 for negatives and positives respectively. Thanks to the novel sorting objective, RS Loss can assign a gradient to suppress a positive example when it is not ranked among positives accordingly wrt its continuous label (e.g. $i = 1$).

S2.1. Analysis to Determine Design Choices in Localisation Loss

In this section, we provide our analysis on ATSS [21] to determine our design choices for localisation. First, as a baseline, we train ATSS network with the following loss function:

$$\mathcal{L}_{RS-ATSS} = \mathcal{L}_{RS} + \lambda_{box}\mathcal{L}_{box}, \tag{S17}$$

where $\mathcal{L}_{RS}$ is our Rank & Sort Loss, λ_{box} is the task-level balancing coefficient and $\mathcal{L}_{box}$ is the box regression loss.

First, we investigate two tuning-free heuristics to determine λ_{box} every iteration: (i) value-based: $\lambda_{box} = \mathcal{L}_{RS}/\mathcal{L}_{box}$, and (ii) magnitude-based: $\lambda_{box} = \left|\frac{\partial \mathcal{L}_{\text{RS}}}{\partial \hat{\mathbf{s}}}\right| / \left|\frac{\partial \mathcal{L}_{\text{box}}}{\partial \hat{\mathbf{b}}}\right|$ where $|\cdot|$ is L1 norm, $\hat{\mathbf{b}}$ and $\mathbf{s}$ are box regression and classification head outputs respectively. Table S1 presents that value-based task balancing performs similar to tuning λ_{box} (~ 0 AP on average).

Table S1. Comparison of instance- and task-level weighting methods on RS-ATSS. Instance-level importance weighting methods yield similar performance and value-based SB achieves similar performance with constant weighting. Thus, we use score-based weighting and value-based SB with RS Loss (underlined&bold), which are both tuning-free.

Instance-level importance weight (w^i)	Task-level balancing coefficient (λ_r)				
	Constant weighting			Self-balance (SB)	
	1	2	3	value	magnitude
No prioritization	38.9	39.7	39.7	39.7	39.4
Centerness-based [18]	38.8	39.8	39.6	39.6	39.5
Score-based [7]	39.1	39.8	39.7	**39.9**	39.7
IoU-based [6]	39.0	39.7	39.8	39.7	39.6
Ranking-based [13]	39.1	**39.9**	39.6	**39.9**	39.6

Secondly, we delve into $\mathcal{L}_{box}$, which is defined as the weighted average of the individual losses of examples:

$$\mathcal{L}_{box} = \sum_{i \in \mathcal{P}} \frac{w^i}{\sum_{j \in \mathcal{P}} w^j} \mathcal{L}_{GIoU}(\hat{b}_i, b_i), \tag{S18}$$

where $\mathcal{L}_{GIoU}(\hat{b}_i, b_i)$ is the GIoU Loss [16], and w^i is the *instance-level importance weight*. Unlike *no prioritization* (i.e. $w^i = 1$ for $i \in \mathcal{P}$), recently, a diverse set of heuristics assigns different importances over $i \in \mathcal{P}$: *centerness-based* importance [18, 21] aims to focus on the proposals (i.e. point or anchor) closer to the center of b_i, *score-based* heuristic [7] uses the maximum of confidence scores of a prediction as w^i, *IoU-based* approach [6] increases the losses of the predictions that are already better localized by $w^i = \text{IoU}(\hat{b}_i, b_i)$, and finally *ranking-based* weighting [13] uses $w^i = \frac{1}{|\mathcal{P}|}\left(\sum_{k \in \mathcal{P}} \frac{\text{H}(x_{ki})}{\text{rank}(k)}\right)$, where $\text{H}(\cdot)$ can be smoothed by an additional hyper-parameter (δ_{loc}). Note that these instance-level weighting methods perform similarly (largest gap is 0.2 AP – Table S1) and we prefer score-based weighting with RS Loss.

S2.2. A Comparative Analysis with aLRP Loss

In this section, we list our observations on aLRP Loss [13] based on our comparative analysis with RS Loss:

Observation 1: Tasks competing with each other within the bounded range of aLRP Loss degrades performance especially when the models are trained 12 epochs following the common training schedule.

To illustrate this, we train Faster R-CNN [15] with aLRP Loss and our RS Loss using two different settings:

- "Standard Training", which refers to the common training (e.g. [15, 18, 21]): The network is trained by a batch size of 16 images with resolution 1333×800 without any augmentation except the standard horizontal flipping. We use 4 GPUs, so each GPU has 4 images during training. We tune the learning rate of aLRP

Table S2. Due to epoch-based self-balance and competition of tasks for the limited range, aLRP Loss performs significantly worse in the first epoch. When the model is trained longer using heavy training (i.e. 100 epochs, SSD-style augmentation [12]), the default configuration of aLRP Loss, the performance gap relatively decreases at the end of training, however, the gap is still significant ($\sim$ 2 AP) for the standard training (i.e. 12 epochs, no SSD-style augmentation [12]). All experiments are conducted on Faster R-CNN.

Loss Function	Heavy Training		Standard Training	
	Epoch 1	Epoch 100	Epoch 1	Epoch 12
aLRP Loss [13]	9.4	40.7	14.4	37.4
RS Loss	**17.7**	**41.2**	**22.0**	**39.6**

Loss as 0.009 and for our RS Loss we set it to 0.012. Consistent with the training image size, the test image size is 1333×800.

- "Heavy Training", which refers to the standard training design of aLRP Loss (and also AP Loss): The network is trained by a batch size of 32 images with resolution 512×512 on 4 GPUs (i.e. 8 images/GPU) using SSD-syle augmentation [12] for 100 epochs. We use the initial learning rate of 0.012 for aLRP Loss as validated in the original paper, and for our RS Loss, we simply use linear scheduling hypothesis and set it to 0.024 without further validation. Here, following aLRP Loss (and AP Loss), the test image size is 833×500.

Table S2 presents the results and we observe the following:

1. *For both "heavy training" and "standard training", aLRP Loss has significantly lower performance after the first epoch (*17.7 *AP vs.* 9.2 *AP for heavy training and* 22.0 *AP vs.* 14.4 *AP) compared to RS Loss*: aLRP Loss has a bounded range between 0 and 1, which is dominated by the classification head especially in the beginning of the training, and hence, the box regression head is barely trained. To tackle that, Oksuz et al. [13] dynamically promotes the loss of box regression head using a self-balance weight, initialized to 50 and updated based on loss values at the end of every epoch. However, we observed that this range pressure has an adverse effect on the performance especially at the beginning of the training, which could not be fully addressed by self-balance since in the first epoch the SB weight is not updated.

2. *While the gap between RS Loss and aLRP Loss is* 0.5 *AP for "heavy training", it is* 2.2 *AP for "standard training"*. After the SB weight of aLRP Loss is updated, the gap can be reduced when the models are trained for longer epochs. However, the final gap is still large ($\sim$ 2 AP) for "standard training" with 12 epochs since unlike aLRP Loss, our RS Loss (i) does not have a single bounded range for which multiple tasks compete, and (ii) uses an iteration-based self-balance instead of epoch-based.

Observation 2: The target of aLRP Loss does not have an intuitive interpretation.

Self-balance (or range pressure – see Observation 1) is not the single reason why RS Loss performs better than aLRP Loss in both scheduling methods in Table S2. aLRP Loss uses the following target error for a positive example i:

$$\ell^*_{aLRP}(i) = \frac{\mathcal{E}_{loc}(i)}{\text{rank}(i)}, \tag{S19}$$

where

$$\mathcal{E}_{loc}(i) = \frac{1 - \text{IoU}(\hat{b}_i, b_i)}{1 - \tau}, \tag{S20}$$

and τ is the positive-negative assignment threshold. However, unlike the target of RS Loss for specifying the error at the target ranking where positives are sorted wrt their IoUs (see Fig. S2), the target of aLRP Loss does not have an intuitive interpretation.

Observation 3: Setting τ in Eq. S20 to the value of the positive-negative (anchor IoU) assignment threshold creates ambiguity (e.g. anchor-free detectors do not have such a threshold).

We identify three obvious reasons: (i) Anchor-free methods do not use IoU to assign positives and negatives, (ii) recent SOTA anchor-based methods, such as ATSS [21] and PAA [6], do not have a sharp threshold to assign positives and negatives, but instead they use adaptive thresholds to determine positives and negatives during training, and furthermore (iii) anchor-based detectors split anchors as positives and negatives; however, the loss is computed on the predictions which may have less IoU with ground truth than 0.50. Note that our RS Loss directly uses IoUs as the continuous labels without further modifying or thresholding them.

Observation 4: Using an additional hyper-parameter (δ_{loc}) for ranking-based weighting yields better performance for the common 12 epoch training.

As also discussed in Section S2.1, *ranking-based* importance weighting of the instances corresponds to:

$$w^i = \frac{1}{|\mathcal{P}|} \left(\sum_{k \in \mathcal{P}} \frac{\text{H}(x_{ki})}{\text{rank}(k)} \right). \tag{S21}$$

aLRP Loss, by default, prefers not to smooth the nominator ($\text{H}(x_{ki})$) but $\text{rank}(k)$ is computed by the smoothed unit-step function. We label this setting as "default" and introduce an additional hyper-parameter δ_{loc} to further analyse

Table S3. Using an additional δ_{loc} to smooth the effect of ranking-based weighting can contribute to the performance.

δ_{loc}	0.00	0.50	1.00	1.50	2.00	Default
AP	39.3	39.4	39.8	**39.9**	39.8	39.5

Table S4. We set $\delta_{RS} = 0.50$, the only hyper-parameter of RS Loss, in all our experiments.

δ_{RS}	0.25	0.40	0.50	0.60	0.75	1.00
AP	39.0	39.7	**39.9**	39.7	39.8	39.4

Table S5. RS Loss improves strong baseline Cascade R-CNN [1].

Method	AP ↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓
Cascade R-CNN	40.3	58.6	44.0	67.0
RS Cascade R-CNN	**41.3**	**58.9**	**44.7**	**66.6**

ranking-based weighting. Note that the larger δ_{loc} is, the less effect the logits will have on w^i (Eq. S18). In Table S3, we compare these different settings on RS-ATSS trained for 12 epochs with our RS Loss, and observe that the default ranking-based weighting can be improved with different δ_{loc} values. However, for our RS Loss, we adopt score-based weighting owing to its tuning-free nature.

S3. More Experiments on RS Loss

This section presents the experiments that are omitted from the paper due to space constraints.

S3.1. Effect of δ_{RS}, the Single Hyper-parameter, for RS Loss.

Table S4 presents the effect of δ_{RS} on RS Loss using ATSS. We observe similar performance between $\delta_{RS} = 0.40$ and $\delta_{RS} = 0.75$. Also note that considering positive-to-positive errors in the sorting error, we set δ_{RS} different from AP Loss and aLRP Loss, both of which smooth the unit step function by using $\delta_{RS} = 1.00$ as validated by Chen et al. [2].

S3.2. Training Cascade R-CNN [1] with RS Loss

Table S5 shows that using RS Loss to train Cascade R-CNN (RS-Cascade R-CNN) also improves baseline Cascade R-CNN by 1.0 AP. We note that unlike the conventional training, we do not assign different loss weights over each R-CNN.

S3.3. Hyper-parameters of R-CNN Variants in Table 1 of the Paper

A two-stage detector that uses random sampling and does not employ a method to adaptively set λ_t^k has at least 7 hyper-parameters since (i) for random sampling, one needs to tune number of foreground examples and number of background examples to be sampled in both stages (4 hyper-parameters), and (ii) at least 3 λ_t^ks need to be tuned as the task-balancing coefficients in a loss with four components (Eq. 1 in the paper). As a result, except aLRP Loss and our RS Loss, all methods have at least 7 hyper-parameters. When the box regression losses of RPN and R-CNN are L1 Loss, GIoU Loss or AutoLoss, and the network has not an additional auxiliary head, 7 hyper-parameters aree sufficient (i.e. GIoU Loss [16], Carafe FPN [19] and AutoLoss-A [11]). Below, we list the methods with more than 7 hyper-parameters:

- FPN [8] uses Smooth L1 in both stages, resulting in 2 more additional additional hyper-parameters (β) to be tuned for the cut-off from L2 Loss to L1 Loss for Smooth L1 Loss.
- IoU-Net [5] also has Smooth L1 in both stages. Besides, there is an additional IoU prediction head trained also by Smooth L1, implying λ_t^k for IoU prediction head and β for Smooth L1. In total, there are 7 hyper-parameters in the baseline model, and with these 4 hyper-parameters, IoU-Net includes 11 hyper-parameters.
- To train R-CNN, Libra R-CNN [14] uses IoU-based sampler, which splits the negatives into IoU bins with an IoU interval width of κ, then also exploits random sampling. Besides it uses Balanced L1 Loss which adds 2 more hyper-parameters to Smooth L1 Loss (3 hyper-parameters in total). As a result, Libra R-CNN has 11 hyper-parameters in sampling and loss function in total.
- Dynamic R-CNN [20] uses Smooth L1 for RPN and adds one more hyper-parameter to the Smooth L1, resulting in 3 additional hyperparameters. As a result, it has 10 hyper-parameters.

S3.4. Using Different Localisation Qualities as Continuous Labels to Supervise Instance Segmentation Methods

In order to provide more insight regarding the employment of continuous labels for the instance segmentation methods, we train YOLACT under four different settings: (i) without using continuous labels (c.f. "Binary" in Table S6) (ii) using IoU, the bounding box quality, as the continuous label (iii) using Dice coefficient, the segmentation quality, as the continuous label and (iv) using the average of IoU and Dice coefficient as the continuous label. Table S6 suggests that all of these localisation qualities improve performance against ignoring them during training. Therefore, we use IoU as the continuous ground truth labels in all of our experiments with the exception of RS-SOLOv2, in which we used Dice coefficient, yielding similar performance to using IoU (Table S6), since SOLOv2 does not have a box regression head.

Table S6. Analysis whether using continuous labels is useful for instance segmentation. We use IoU to supervise instance segmentation methods except SOLOv2, in which we use Dice coefficient since bounding boxes are not included in the output. Using Dice coefficient also provides similar performance with IoU. Binary refers to the conventional training (i.e. only ranking without sorting) without continuous labels.

Label	Segmentation			Detection		
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
Binary	29.1	49.9	29.4	32.9	53.8	34.2
IoU	**29.9**	**50.5**	**30.6**	**33.8**	54.2	**35.4**
Dice	29.8	50.4	30.2	33.5	**54.3**	35.1
(IoU+Dice)/2	29.6	50.2	30.0	33.4	54.1	34.8

S3.5. Details of the Ablation Analysis on Different Degrees of Imbalance

This section presents details on the discussion on robustness of RS Loss to imbalance (Section 6.3 in the paper).

Experimental Setup. Using RS Loss on multi-stage visual detectors (e.g. Faster R-CNN or Mask R-CNN) involves two major changes in the training pipeline:

1. The random samplers from both stages (i.e. from RPN and R-CNN) are removed.

2. The $O + 1$-way softmax classifier, where O is the number of classes in the dataset, is replaced by O binary (i.e. class-wise) sigmoid classifiers for the second stage of Faster R-CNN (i.e. R-CNN)[1].

Note that in order to present the actual imbalance ratio between positives (pos) and negatives (neg), one needs to track the actual *task* ratio resulting from the binary sigmoid classifiers. That is, with O individual binary sigmoid classifiers, each positive instance (e.g. anchor, proposal/region-of-interest) yields 1 pos and $O - 1$ neg tasks, and each negative instance yields O negative tasks (also refer to Section 3.1 of Tan et al. [17] for details). To illustrate (Table S7), when we aim 1:3 pos:neg instance ratio for R-CNN by using a random sampler, as conventionally done, the actual *instance pos:neg ratio* turns out to be 1:8 since the sampler pads the fixed batch size (i.e. in terms of proposals/regions-of-interest, which is 256 in this case) with negative instances when there is no enough positives. On the other hand, the actual *task pos:neg ratio* is 1:702, implying that the pos:neg ratio of instances is not representative. As a result, we consider the actual task pos:neg ratio as the actual imbalance ratio.

Robustness of RS Loss to Imbalance. In order to show that RS Loss is robust to different degrees of imbalance without tuning, we trained (i) three Faster R-CNN [15] on COCO dataset [10] by gradually removing the random sampler from both stages and also (ii) one Mask R-CNN on LVIS dataset [4] as an extremely imbalanced case. Table S7 presents pos:neg instance and task ratios averaged over the iterations during the first epoch[2]:

- When the random samplers are removed from both stages, the actual pos:neg task ratio increases. Specifically, due to the large number of anchors used for training RPN, actual pos:neg task ratio increases significantly for RPN (from 1:7 to 1:6676). As for R-CNN, this change is not as dramatic as RPN on COCO dataset after the sampler is removed (from 1:702 to 1:1142 – compare "Random" and "None" for R-CNN in Table S7) since R-CNN is trained with top-1000 scoring region-of-interests (instead of all anchors in RPN) and COCO dataset has 80 classes. Note that RS Loss can train all three configurations (whether random sampling is removed or not) for COCO dataset successfully, and when more data is available (i.e. sampler is "None"), the performance improves from 38.5 to 39.6.

- When we train Mask R-CNN using RS Loss on the long-tailed LVIS dataset without any samplers, we observed that unlike COCO dataset, R-CNN training is *extremely imbalanced* (actual pos:neg task ratio is 1:10470) due to the large number of classes in LVIS dataset. Still, our RS Loss achieves SOTA performance despite this extreme imbalance (see also Table 5 in the paper).

As a result, we conclude that RS Loss can easily be incorporated to train data with different levels of imbalance.

Are Score-based Loss Functions Robust to Imbalance Without Tuning? Here, we investigate how Cross-entropy Loss and Focal Loss behave when samplers are removed without any tuning.

Cross-entropy Loss. As a fair baseline for our RS Loss, we use Faster R-CNN with GIoU Loss and only remove random sampling gradually similar to how we did for RS Loss. Table S8 shows that, as opposed to our RS Loss, the performance significantly drops once the samplers are removed for Cross-entropy Loss, and hence Cross-entropy Loss cannot be directly employed to train different levels of imbalance unlike our RS Loss.

Focal Loss. There are many design choices that one needs to tune in order replace the standard Cross-entropy Loss by Focal Loss. Here, instead of tuning each of these

[1]Note that RPN, which aims to determine "objectness", is already implemented by a single sigmoid classifier in mmdetection [3]. Hence, no modification is required for the classifier of RPN.

[2]Note that since the anchors, fed to the first stage (i.e. RPN), are fixed in location, scale and aspect ratio during the training, the imbalance ratios in the first epoch also applies for all epochs for RPN; on the other hand, for R-CNN the number of negatives for each positive may increase in the latter epochs since the RPN will be able to classify and locate more objects.

Table S7. Ablation with different degrees of imbalance. Positive:negative (pos:neg) ratios averaged over the iterations of the first epoch of training with RS Loss on different datasets & samplers. For each pos instance, e.g. anchor, random sampler aims to sample 1 neg instance in RPN and 3 neg instance in R-CNN. With no sampler (None), there is no desired pos:neg instance ratio, i.e. not available - N/A. Note that when random sampler cannot find enough pos, the batch is padded with neg; accordingly, actual pos:neg instance ratio is computed after this padding. Since we use binary sigmoid classifiers on R-CNN; each neg instance has O neg tasks, and each pos instance has 1 pos and $O-1$ neg tasks where O is the number of dataset classes. Using this, actual pos:neg task ratio (underlined) presents the actual imbalance ratio in the input of RS Loss. Quantitatively, the actual pos:neg task ratio varies between 1:7 to 1:10470. Despite very different degrees of imbalance, our RS Loss outperforms all counterparts without sampling and tuning in both datasets.

Dataset	Sampler		desired pos:neg instance ratio		actual pos:neg instance ratio		actual pos:neg task ratio		AP
	RPN	R-CNN	RPN	R-CNN	RPN	R-CNN	RPN	R-CNN	
COCO	Random	Random	1:1	1:3	1:7	1:8	1:7	1:702	38.5
	None	Random	1:1	N/A	1:6676	1:8	1:6676	1:702	39.3
	None	None	N/A	N/A	1:6676	1:13	1:6676	1:1142	39.6
LVIS	None	None	N/A	N/A	1:3487	1:12	1:3487	1:10470	25.2

Table S8. RS Loss is robust to class imbalance while score-based loss functions cannot handle imbalanced data in this case. RS Loss successfully trains Faster R-CNN with both relatively balanced ("Random" sampling) and severely imbalanced ("None" in the table) data. Numbers in parentheses show positive to negative ratio of sampled examples.

Loss	RPN	R-CNN	AP	AP_{50}	AP_{75}
Cross-entropy	Random	Random	37.6	58.2	41.0
	None	Random	32.7	49.4	35.9
	None	None	30.1	45.3	33.2
Focal Loss [9]	Random	Random	31.8	47.4	35.0
	None	Random	32.0	47.5	35.1
	None	None	30.7	44.6	34.2
RS Loss (Ours)	Random	Random	38.5	58.5	41.5
	None	Random	39.3	**59.6**	42.3
	None	None	**39.6**	59.5	**43.0**

extensively, we use a commonly used setting in one-stage detectors [7, 9, 21] to train RPN and R-CNN. In particular, we use individual class-wise binary sigmoid classifier (as we also did for RS Loss), set the learning rate to 0.01, the weight of GIoU Loss to 2 and the bias terms in the last layer of the classification head[3] such that the confidence scores of the positives are 0.01 to prevent destabilization of the training due to large loss value originating from negatives. However, we observed that Focal Loss is not able to perform as good as Cross-entropy Loss and RS Loss with this configuration (Table S7). Hence, as a generalisation of Cross-entropy Loss, Focal Loss at least needs to be tuned carefully in order to yield better performance.

As a result, we conclude that common score-based loss functions (i.e. Cross-entropy Loss and Focal Loss) cannot handle different degrees of imbalance without tuning; while our RS Loss can.

[3]Note that we also do not tune this bias term for our RS Loss and use the default setting for all the detectors that we train.

S3.6. Effect of RS Loss on Efficiency

We discuss the effect on efficiency on two levels: (i) training and (ii) inference.

S3.6.1 Effect on Training Efficiency

Similar to other ranking based loss functions (i.e. AP Loss [2] and aLRP Loss [13]), RS Loss has also quadratic time complexity, and as a result, one training iteration of RS Loss takes 1.5× more on average (Table S9).

S3.6.2 Effect on Inference Efficiency

We observed that the methods trained by RS Loss yield larger confidence scores than the baseline methods, which are trained by score-based loss functions (e.g. Cross-entropy Loss). As a result, for inference efficiency, the score threshold to discard detections associated with background before Non-Maximum Suppression (NMS) should be set carefully[4]. Here, we provide examples using multi-stage visual detectors on two datasets:

- **COCO dataset.** Faster R-CNN and Mask R-CNN use 0.05 as the confidence score threshold on COCO dataset when they are trained by Cross-entropy Loss, that is, all the detections with confidence score less than 0.05 are regarded as background (i.e. false positive), and they are simply removed from the detection set before NMS. Keeping this setting as 0.05, RS-R-CNN with ResNet-50 reaches 39.6 AP and 67.9 oLRP but with slower inference time than the baseline Cross-entropy Loss, which has 21.4 fps. Then, tuning this score threshold to 0.40, Faster R-CNN trained by our RS Loss performs exactly the same (see 39.6 AP and 67.9 oLRP in Table S9) at 22.5 fps, slightly faster than the baseline Faster R-CNN. Table S9 presents the results on Faster R-CNN and Mask R-CNN with the

[4]We keep the default settings of the methods in the paper

Table S9. Average iteration time of methods trained by the standard loss vs. RS Loss. On average, training with RS Loss incurs $\sim 1.5\times$ longer time due to its quadratic time complexity similar to other existing ranking-based loss functions [2, 13].

Method	Standard Loss (sec)	RS Loss (sec)
Faster R-CNN	0.42	0.82
Cascade R-CNN	0.51	2.26
ATSS	0.44	0.70
PAA	0.57	0.99
Mask R-CNN	0.64	1.04
YOLACT	0.57	0.59
SOLOv2-light	0.64	0.90

tuned confidence score threshold, that is 0.40. While the performance of models in Table S9 in terms of oLRP is always equal to the ones with confidence score of 0.05, in some rare cases we observed negligible performance drop (i.e. up to 0.01 AP points, e.g. RS Faster R-CNN+ drops from 40.8 AP to 40.7 AP).

- **LVIS dataset.** Table S11 presents the results of Mask R-CNN on LVIS dataset. Similar to COCO dataset, when we use RS Loss, we prefer a larger confidence score threshold, that is 0.60, and also we observe that RS Loss is robust to this threshold choice, while the performance of the standard Mask R-CNN degrades rapidly when the score threshold increases. As a result, when the score threshold is set accordingly, our RS-Mask R-CNN yields 25.2 AP at 11.7 fps, which outperforms the baseline Mask R-CNN with 21.7 AP at 11.0 fps in the best confidence score setting.

As a result, the models trained by our RS Loss outputs larger confidence scores, and accordingly, the score threshold needs to be adjusted accordingly for better efficiency.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 5

[2] Kean Chen, Weiyao Lin, Jianguo li, John See, Ji Wang, and Junni Zou. Ap-loss for accurate one-stage object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1–1, 2020. 2, 5, 7, 8

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv*, 1906.07155, 2019. 6

[4] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[5] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *The European Conference on Computer Vision (ECCV)*, 2018. 5

[6] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. In *The European Conference on Computer Vision (ECCV)*, 2020. 3, 4

[7] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3, 7

[8] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[9] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(2):318–327, 2020. 7

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *The European Conference on Computer Vision (ECCV)*, 2014. 6

[11] Peidong Liu, Gengwei Zhang, Bochao Wang, Hang Xu, Xiaodan Liang, Yong Jiang, and Zhenguo Li. Loss function discovery for object detection via convergence-simulation driven search. In *International Conference on Learning Representations (ICLR)*, 2021. 5

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *The European Conference on Computer Vision (ECCV)*, 2016. 4

[13] Kemal Oksuz, Baris Can Cam, Emre Akbas, and Sinan Kalkan. A ranking-based, balanced loss function unifying classification and localisation in object detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3, 4, 7, 8

[14] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: Towards balanced learning for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(6):1137–1149, 2017. 3, 6

[16] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 5

[17] Jingru Tan, Xin Lu, Gang Zhang, Changqing Yin, and Quanquan Li. Equalization loss v2: A new gradient balance approach for long-tailed object detection. In *IEEE/CVF*

Table S10. Comprehensive performance results of models trained by RS-Loss on COCO *minival*. We report AP-based and oLRP-based performance measures, an also inference time on a single Tesla V100 GPU as fps. We set NMS score threshold of RS-R-CNN and RS-Mask R-CNN to 0.40 for COCO dataset.

Method	Backbone	Epoch	MS train	AP↑	AP_{50} ↑	AP_{75} ↑	oLRP ↓	$oLRP_{Loc}$ ↓	$oLRP_{FP}$ ↓	$oLRP_{FN}$ ↓	fps
Object Detection											
RS-Faster R-CNN	R-50	12	–	39.6	59.5	43.0	67.9	16.3	27.8	45.4	22.5
RS-Mask R-CNN	R-50	12	–	40.0	59.8	43.4	67.5	16.1	27.6	44.7	22.8
RS-Faster R-CNN+	R-50	12	–	40.7	61.4	43.8	66.9	16.3	26.4	43.7	21.5
RS-Mask R-CNN+	R-50	12	–	41.1	61.4	44.8	66.6	16.0	26.2	44.0	21.1
RS-Mask R-CNN	R-101	36	[640, 800]	44.7	64.4	48.8	63.7	14.7	24.5	41.3	17.1
RS-Mask R-CNN+	R-101	36	[480, 960]	46.1	66.2	50.3	62.6	14.5	23.5	39.9	16.6
RS-Faster R-CNN	R-101-DCN	36	[480, 960]	47.6	67.8	51.2	61.1	14.4	22.7	37.9	14.1
RS-Faster R-CNN+	R-101-DCN	36	[480, 960]	47.6	68.1	51.9	60.9	14.7	20.9	38.2	13.6
RS-Mask R-CNN+	R-101-DCN	36	[480, 960]	48.7	68.9	53.1	60.2	14.3	21.3	36.9	13.5
RS-Mask R-CNN+	X-101-DCN	36	[480, 960]	49.9	70.0	54.3	59.1	14.0	20.3	36.2	6.4
Instance Segmentation											
RS-Mask R-CNN	R-50	12	–	36.4	57.3	39.2	70.1	18.2	28.7	46.5	17.1
RS-Mask R-CNN+	R-50	12	–	37.3	58.6	40.2	69.4	18.1	28.0	45.3	16.7
RS-Mask R-CNN	R-101	36	[640, 800]	40.3	61.9	43.8	66.9	17.3	24.3	43.0	14.8
RS-Mask R-CNN+	R-101	36	[480, 960]	41.4	63.6	44.7	65.9	17.1	22.8	42.2	14.3
RS-Mask R-CNN+	R-101-DCN	36	[480, 960]	43.5	66.5	47.2	64.0	17.2	22.3	38.5	11.9
RS-Mask R-CNN+	X-101-DCN	36	[480, 960]	44.4	67.8	47.7	63.1	17.1	21.1	37.7	6.0

Table S11. The performances of RS-Mask R-CNN and baseline Mask R-CNN (i.e. trained by Cross-entropy Loss) over different confidence score thresholds on LVIS v1.0 val set. RS-Mask R-CNN is robust to score threshold while the performance of Mask R-CNN degrades rapidly especially for rare classes. Our best method achieves 25.2 mask AP at 11.7 fps (bold), which is also slightly faster than the best performing method of Mask R-CNN (underlined). Accordingly, we set NMS score threshold of RS-Mask R-CNN to 0.60 for LVIS dataset. Inference time is reported on a single A100 GPU.

Score threshold	Mask R-CNN						RS-Mask R-CNN					
	AP_{mask}	AP_r	AP_c	AP_f	AP_{box}	fps	AP_{mask}	AP_r	AP_c	AP_f	AP_{box}	fps
10^{-4}	21.7	9.6	21.0	27.8	22.5	3.2	25.2	16.8	24.3	29.9	25.9	0.2
10^{-3}	<u>21.7</u>	<u>9.6</u>	<u>21.0</u>	<u>27.8</u>	<u>22.5</u>	<u>11.0</u>	25.2	16.8	24.3	29.9	25.9	0.2
10^{-2}	21.1	8.3	20.4	27.6	22.0	13.6	25.2	16.8	24.3	29.9	25.9	0.2
0.10	14.7	2.0	11.6	23.6	15.4	21.4	25.2	16.8	24.3	29.9	25.9	0.2
0.40	8.5	0.5	4.4	16.5	8.9	24.0	25.2	16.8	24.3	29.9	25.9	1.2
0.60	6.1	0.4	2.3	12.9	6.4	24.4	**25.2**	**16.8**	**24.3**	**29.9**	**25.9**	**11.7**
0.80	4.0	0.1	1.1	8.8	4.1	24.8	17.0	6.3	14.3	24.7	17.6	21.2

Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 6

[18] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3

[19] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 5

[20] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic r-cnn: Towards high quality object detection via dynamic training. In *The European Conference on Computer Vision (ECCV)*, 2018. 5

[21] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4, 7