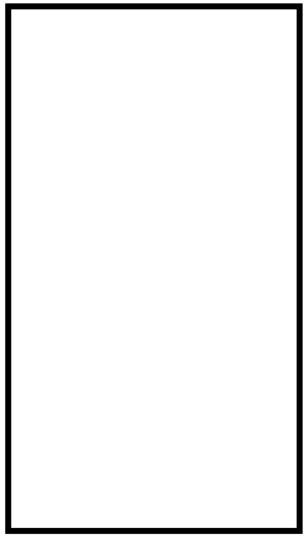
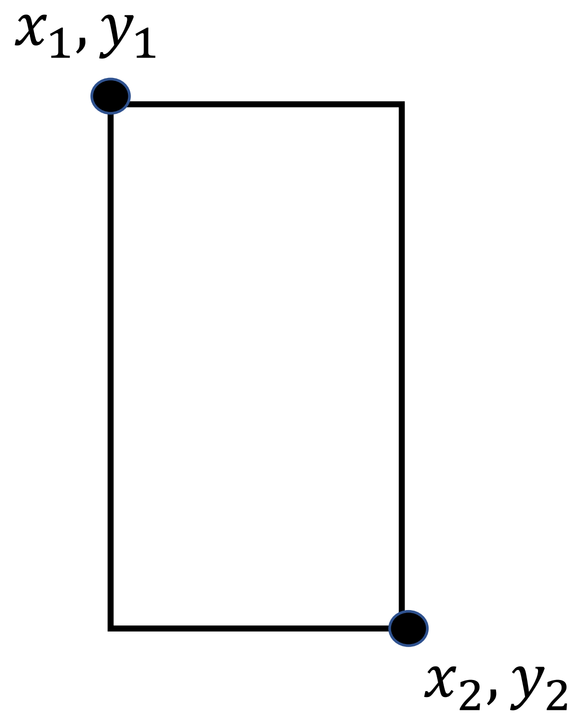
Bounding Box Regression Loss

Bounding Box Parametrization



Bounding Box Parametrization

1. $B = (x_1, y_1, x_2, y_2)$

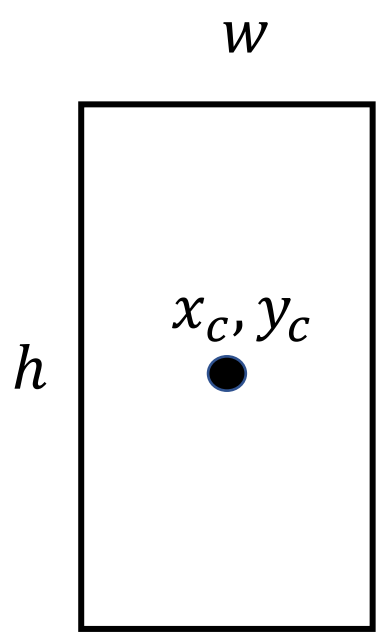


Bounding Box Parametrization

1. $B = (x_1, y_1, x_2, y_2)$

2. $B = (x_c, y_c, w, h)$

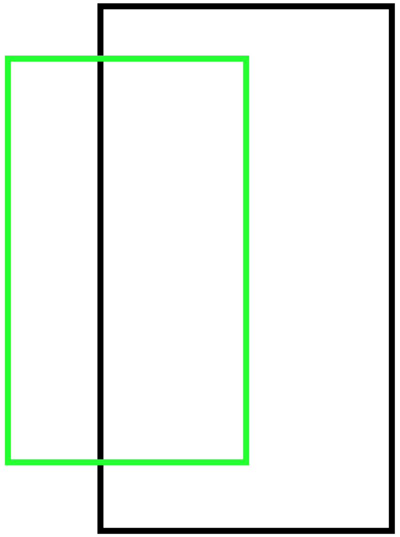
$\vdots$



Bounding Box Regression Loss

Predicted $B^p = \left(x_1^p, y_1^p, x_2^p, y_2^p\right)$

Truth $B^g = \left(x_1^g, y_1^g, x_2^g, y_2^g\right)$



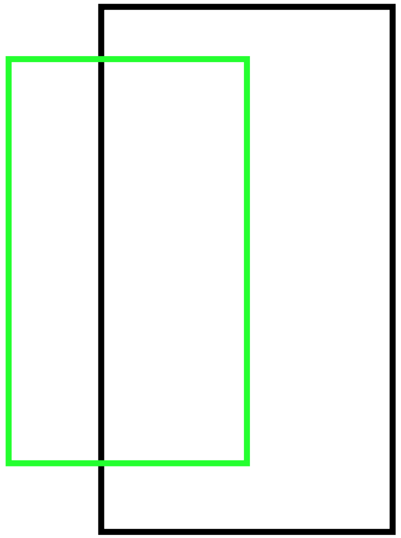
$Loss = MSE(B^p, B^g)$

$Loss = \ell_1 - smooth(B^p, B^g)$

Evaluation Metric

Intersection over Union (*IoU*), known as Jaccard Index

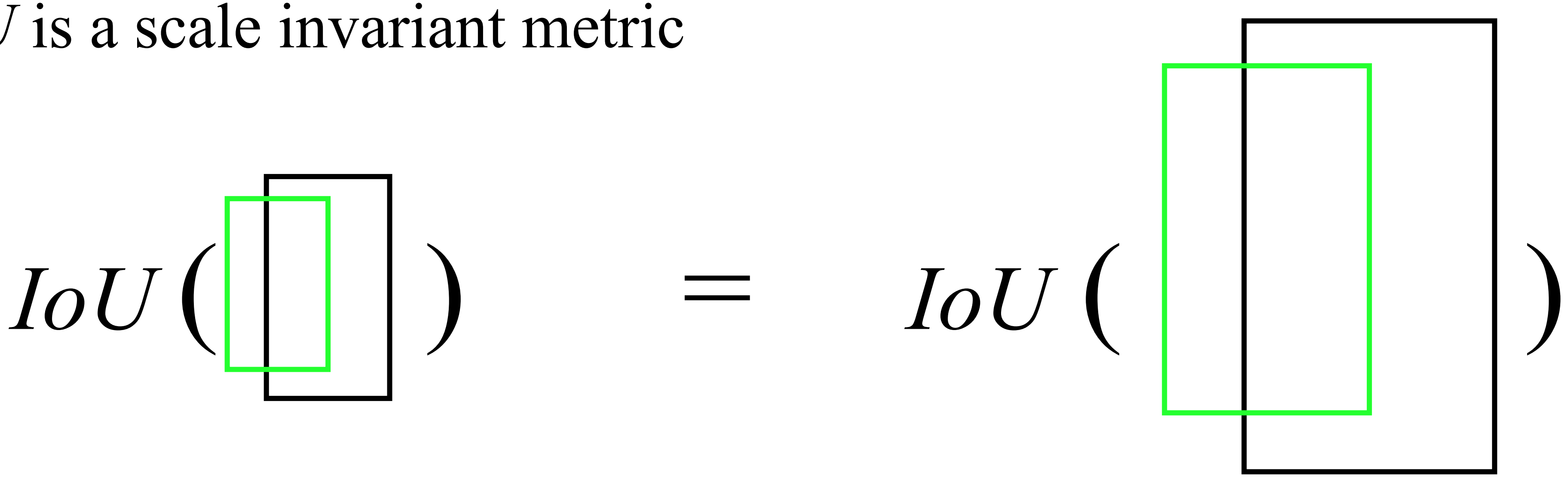
$$IoU = \frac{|A \cap B|}{|A \cup B|}$$



Evaluation Metric

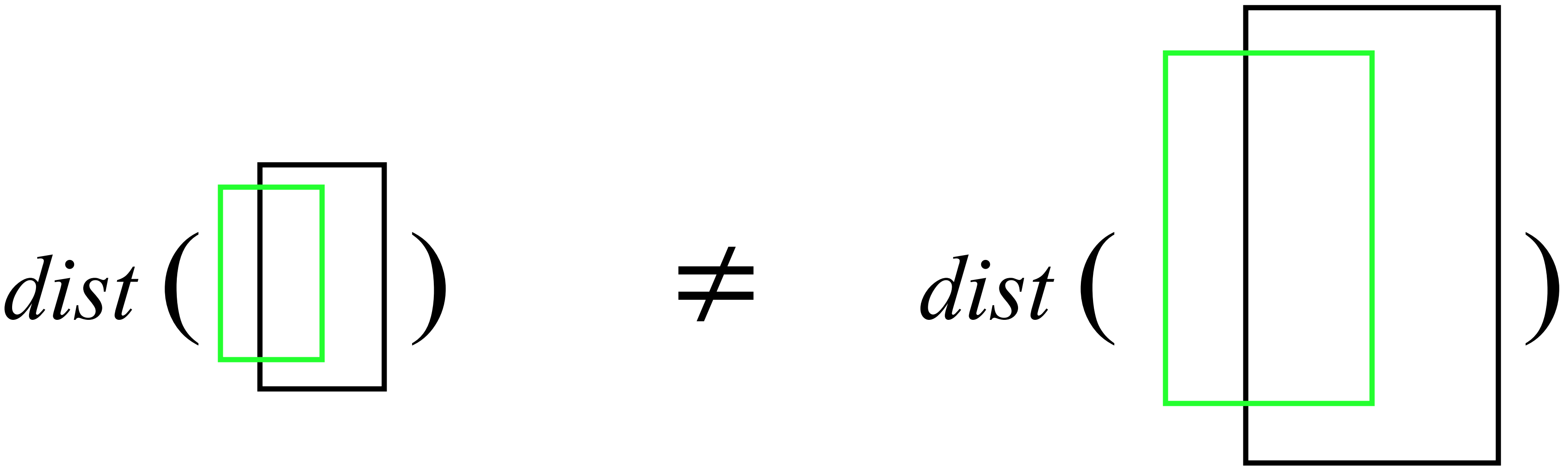
What we like about *IoU*

- Encoding all shape properties into the region property and calculating a normalized measure that focuses on areas (or volumes).
- *IoU* is a scale invariant metric



Loss vs Metric

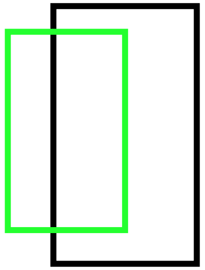
1. *IoU* is invariant to the scale of the problem, but this is not the case for these losses.



Loss vs Metric

1. *IoU* is invariant to the scale of the problem, but this is not the case for these losses.

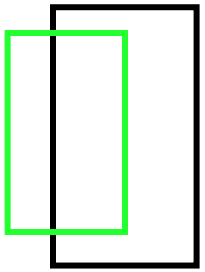
2. A distance loss over different types, of parameters are *e.g.* position, size and angle, is heuristically normalized by regulizers.

$dist\,($  $) = dist_{xy} + \lambda_1 dist_{wh}$

Loss vs Metric

1. *IoU* is invariant to the scale of the problem, but this is not the case for these losses.

2. A distance loss over different types, of parameters are *e.g.* position, size and angle, is heuristically normalized by regulizers.

$dist\,($  $) = dist_{xy} + \lambda_1 dist_{wh} + \lambda_2 dist_{\theta}$

Loss vs Metric

1. *IoU* is invariant to the scale of the problem, but this is not the case for these losses.

2. A distance loss over different types, of parameters are *e.g.* position, size and angle, is heuristically normalized by regulizers.

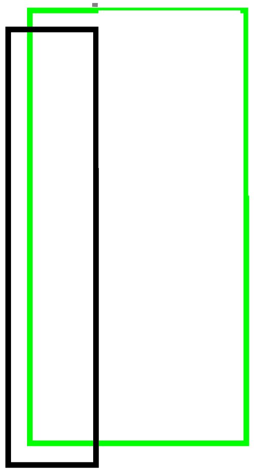
3. There is a weak correlation between minimizing the commonly used regression losses and improving their *IoU* values.

Loss vs Metric

Predicted $B^p = \left(x_1^p, y_1^p, x_2^p, y_2^p\right)$

Truth $B^g = \left(x_1^g, y_1^g, x_2^g, y_2^g\right)$

Loss vs Metric



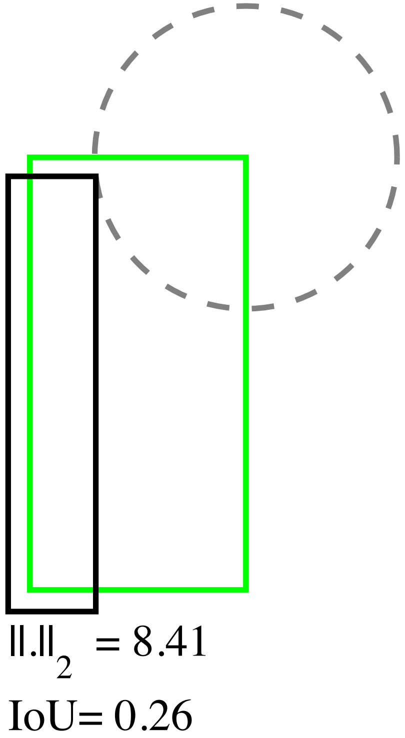
$\|.\|_2 = 8.41$

IoU= 0.26

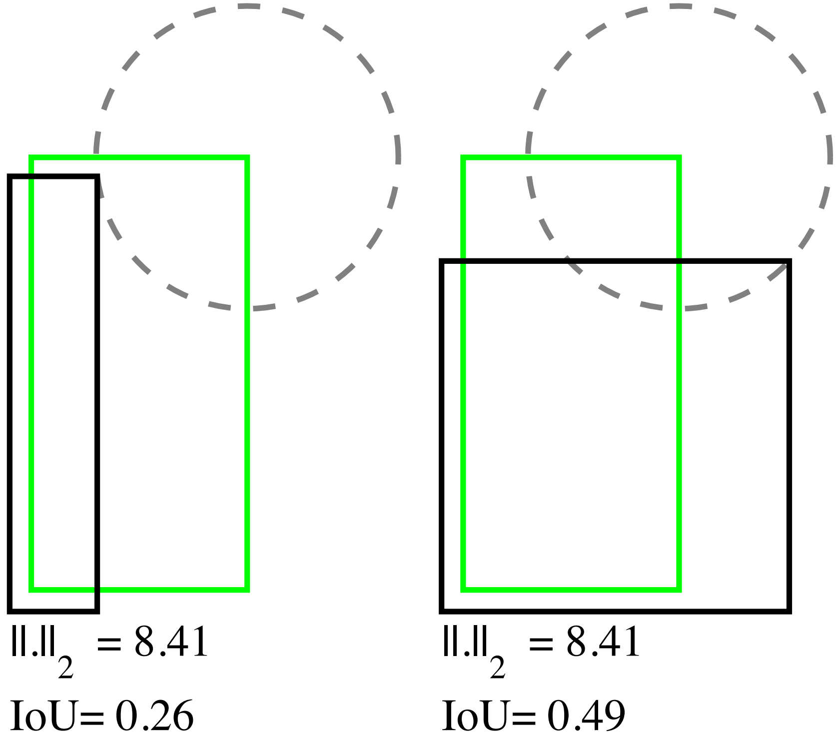
Predicted $B^p = \left(x_1^p, y_1^p, x_2^p, y_2^p\right)$

Truth $B^g = \left(x_1^g, y_1^g, x_2^g, y_2^g\right)$

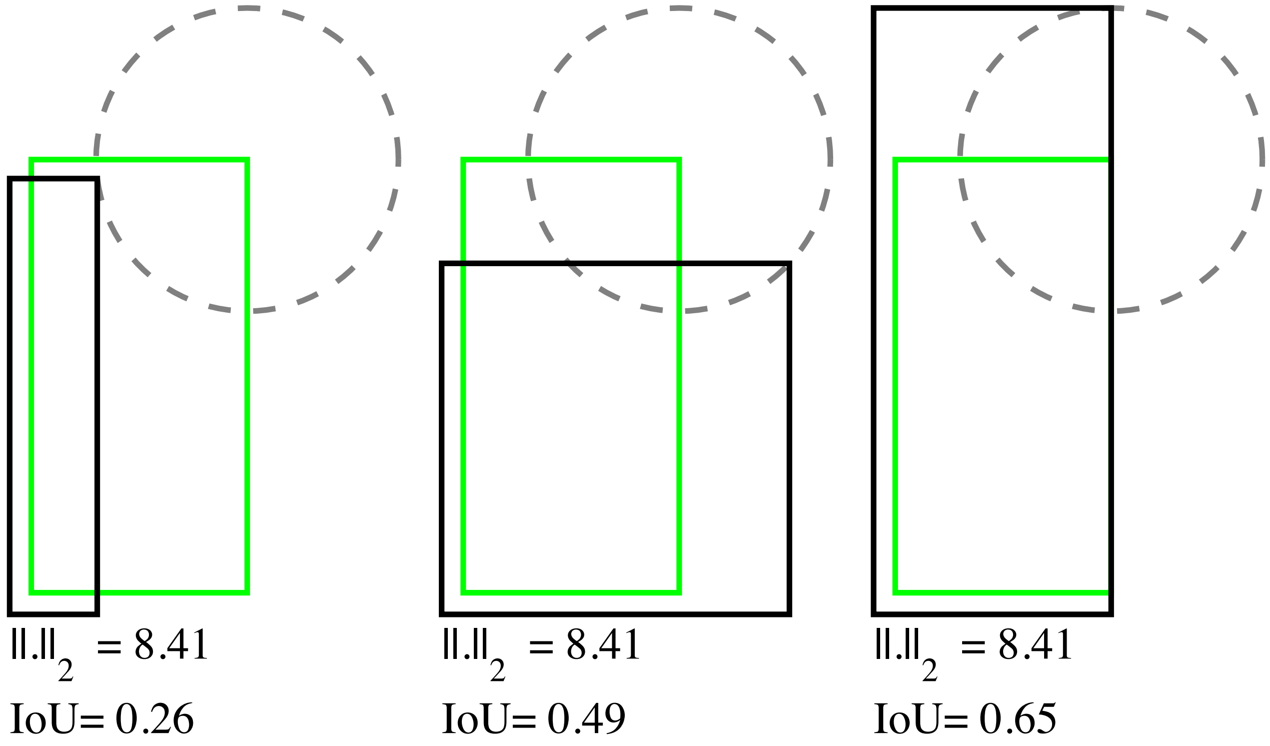
Loss vs Metric



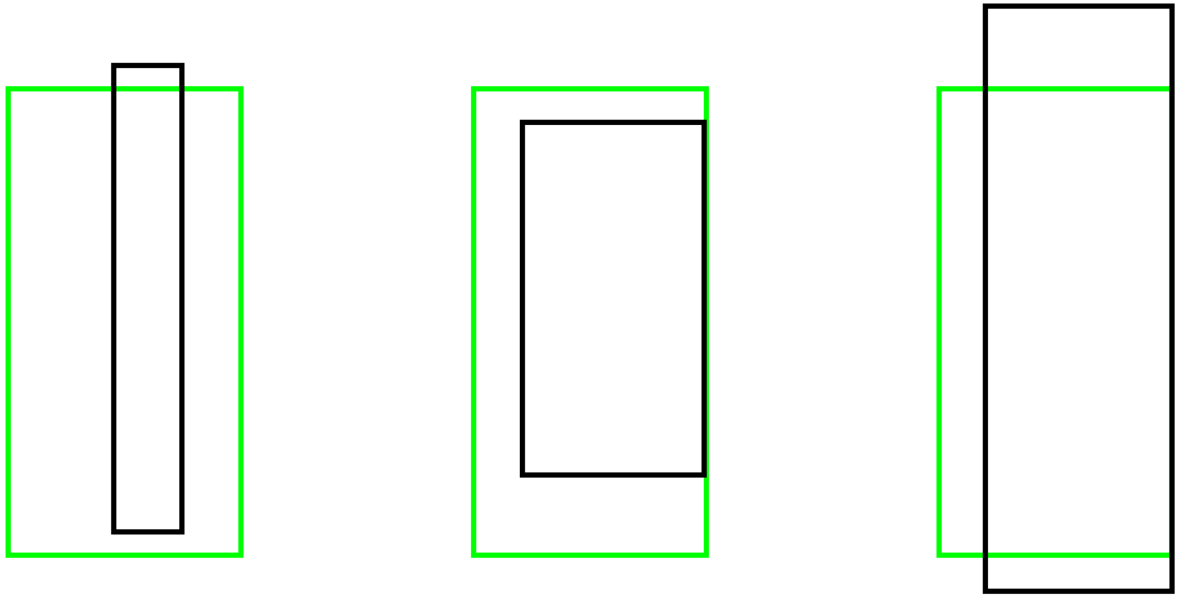
Loss vs Metric



Loss vs Metric



Loss vs Metric



$\|.\|_1 = 9.07$ | $\|.\|_1 = 9.07$ | $\|.\|_1 = 9.07$

IoU = 0.27 | IoU = 0.59 | IoU = 0.66

Predicted $B^p = \left(x_c^p, y_c^p, w^p, h^p\right)$

Truth $B^g = \left(x_c^g, y_c^g, w^g, h^g\right)$

YOLO v1 Regression Loss

loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

2.4. Limitations of YOLO

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.

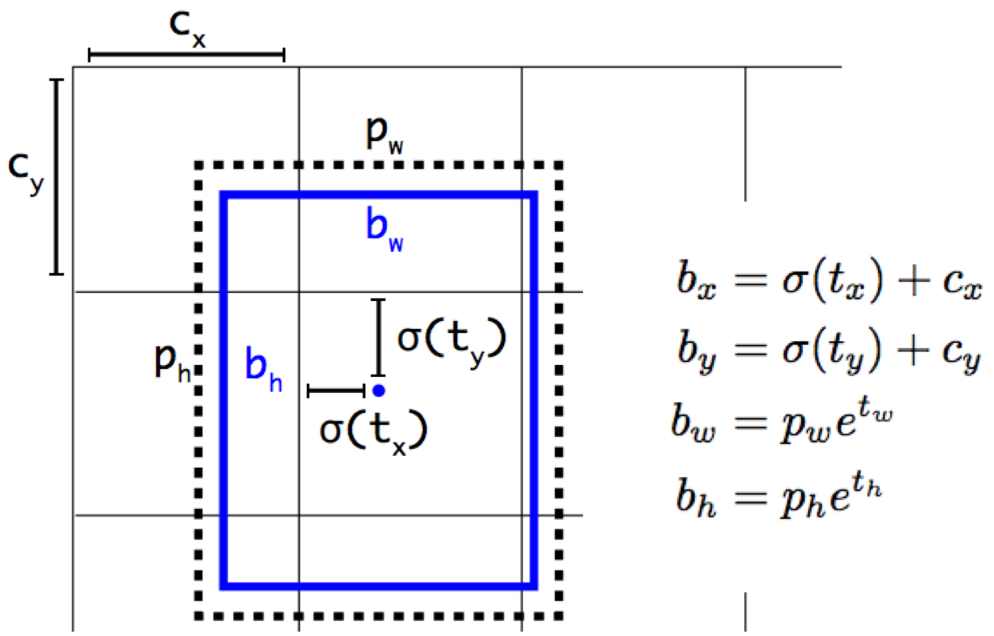
Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. CVPR, 2016.

Faster/Mask R-CNN and YOLO v3 Loss

1. Introducing the concept of an anchor box as a hypothetically good initial guess

2. Using a non-linear representation to naively compensate for the scale change



1. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS, 2015
2. K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. ICCV, 2017
3. J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018

IoU as Loss

The **optimal** objective to optimize for a metric is **the metric** itself.

IoU as Loss

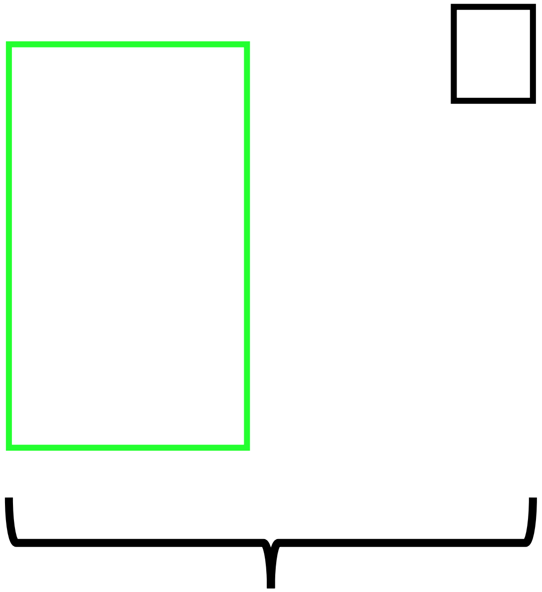
<u>The **optimal** objective to optimize for a metric is **the metric** itself.</u>

In contrast to the prevailing belief, *IoU* between two axis aligned rectangle is **backpropagable** [1].

[1] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. ACM on Multimedia, 2016.

IoU Weakness

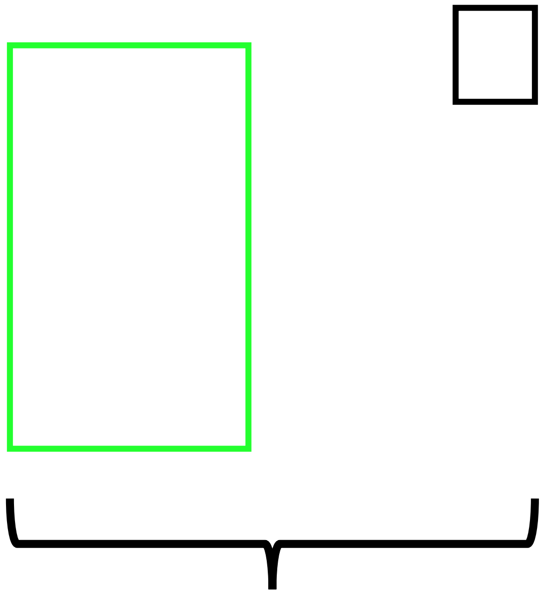
I. if two objects do not overlap, the *IoU* value will be zero



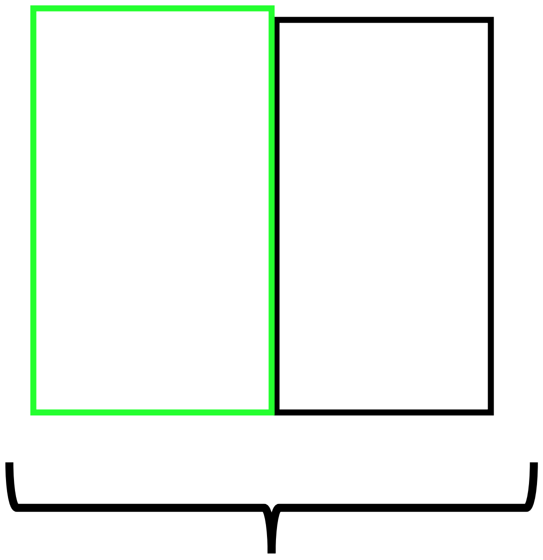
$IoU = 0$

IoU Weakness

I. if two objects do not overlap, the IoU value will be zero



$IoU = 0$

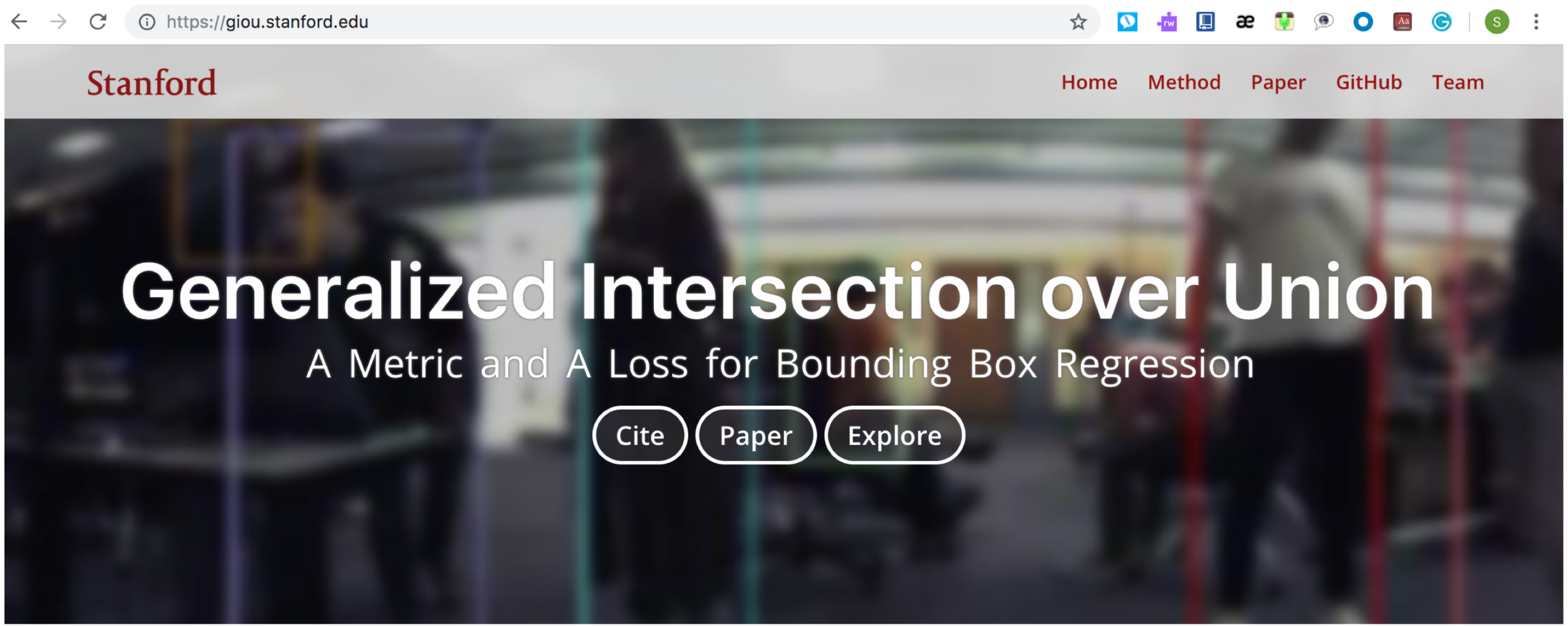


$IoU = 0$

Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression

Hamid Rezatofighi[1,2] Nathan Tsoi[1] JunYoung Gwak[1] Amir Sadeghian[1] Ian Reid[2] Silvio Savarese[1]

[1]Computer Science Department, Stanford University, United States
[2]School of Computer Science, The University of Adelaide, Australia
hamidrt@stanford.edu



Our Contribution

- Introducing the generalized version of *IoU (GIoU)* , as a new metric for comparing any two arbitrary shapes.

Our Contribution

- Introducing the generalized version of *IoU (GIoU)* , as a new metric for comparing any two arbitrary shapes.

- Analytical solution for using *GIoU* as loss between two axis-aligned rectangles or generally n-orthotopes.

Our Contribution

- Introducing the generalized version of *IoU (GIoU)* , as a new metric for comparing any two arbitrary shapes.

- Analytical solution for using *GIoU* as loss between two axis-aligned rectangles or generally n-orthotopes.

- Improving Faster R-CNN, Mask R-CNN and YOLO v3 performance (**%2~%15** relative improvements) on PASCAL VOC and COCO benchmarks.

Our Solution

In this paper, we address the weakness of *IoU* by extending the concept to non-overlapping cases.

Our Solution

In this paper, we address the weakness of *IoU* by extending the concept to non-overlapping cases.

We ensure this generalization:

Our Solution

In this paper, we address the weakness of *IoU* by extending the concept to non-overlapping cases.

We ensure this generalization:

a) follows the same definition as *IoU*, i.e. encoding the shape properties of the compared objects into the region property;

Our Solution

In this paper, we address the weakness of IoU by extending the concept to non-overlapping cases.

We ensure this generalization:

a) follows the same definition as IoU, i.e. encoding the shape properties of the compared objects into the region property;

b) maintains the scale invariant property of IoU, and

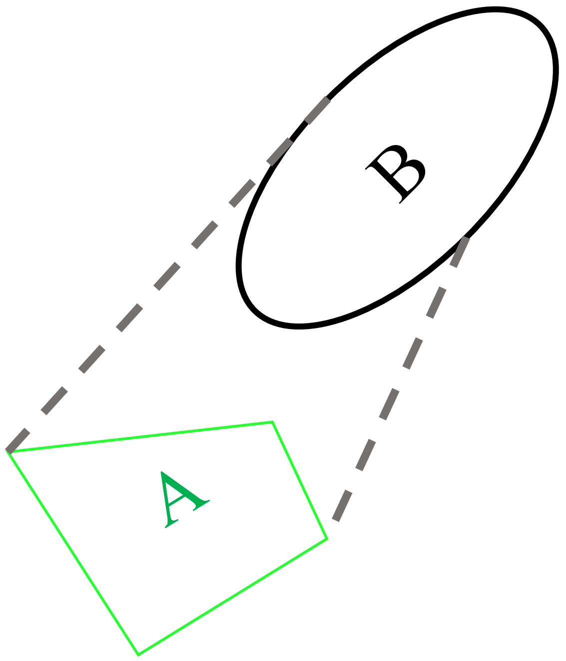
Our Solution

In this paper, we address the weakness of IoU by extending the concept to non-overlapping cases.

We ensure this generalization:

a) follows the same definition as IoU, i.e. encoding the shape properties of the compared objects into the region property;
b) maintains the scale invariant property of IoU, and
c) ensures a strong correlation with IoU in the case of overlapping objects.

GIoU



Algorithm 1: Generalized Intersection over Union

input : Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

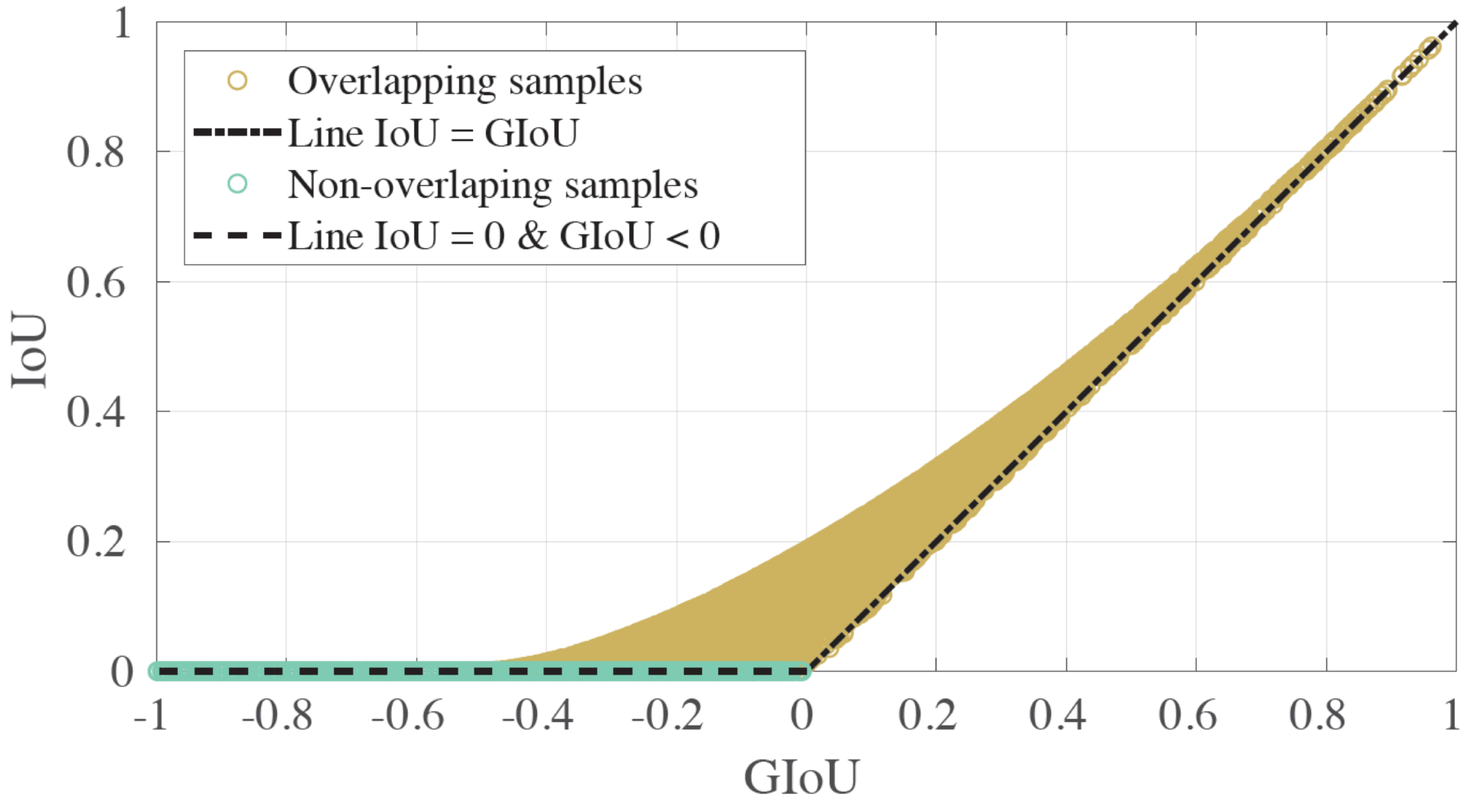
output: $GIoU$

1 For A and B, find the smallest enclosing convex object C, where $C \subseteq \mathbb{S} \in \mathbb{R}^n$

2 $IoU = \dfrac{|A \cap B|}{|A \cup B|}$

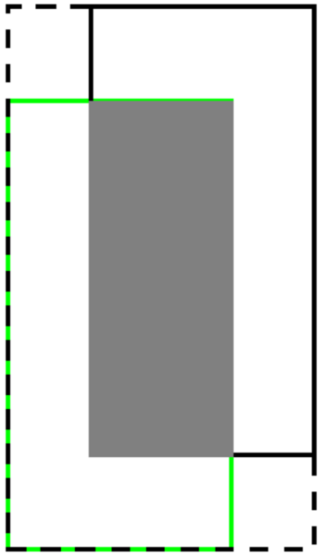
3 $GIoU = IoU - \dfrac{|C \backslash (A \cup B)|}{|C|}$

Correlation between *IoU* and *GIoU*



GIoU as Loss

For two axis aligned rectangle , *GIoU* has a well-behaved derivative



Algorithm 2: IoU and $GIoU$ as bounding box losses

input : Predicted B^p and ground truth B^g bounding box coordinates:
$B^p = (x_1^p, y_1^p, x_2^p, y_2^p)$, $\quad B^g = (x_1^g, y_1^g, x_2^g, y_2^g)$.

output: $\mathcal{L}_{IoU}$, $\mathcal{L}_{GIoU}$.

1 For the predicted box B^p, ensuring $x_2^p > x_1^p$ and $y_2^p > y_1^p$:
$\hat{x}_1^p = \min(x_1^p, x_2^p)$, $\quad \hat{x}_2^p = \max(x_1^p, x_2^p)$,
$\hat{y}_1^p = \min(y_1^p, y_2^p)$, $\quad \hat{y}_2^p = \max(y_1^p, y_2^p)$.

2 Calculating area of B^g: $A^g = (x_2^g - x_1^g) \times (y_2^g - y_1^g)$.

3 Calculating area of B^p: $A^p = (\hat{x}_2^p - \hat{x}_1^p) \times (\hat{y}_2^p - \hat{y}_1^p)$.

4 Calculating intersection $\mathcal{I}$ between B^p and B^g:
$x_1^{\mathcal{I}} = \max(\hat{x}_1^p, x_1^g)$, $\quad x_2^{\mathcal{I}} = \min(\hat{x}_2^p, x_2^g)$,
$y_1^{\mathcal{I}} = \max(\hat{y}_1^p, y_1^g)$, $\quad y_2^{\mathcal{I}} = \min(\hat{y}_2^p, y_2^g)$,

$$\mathcal{I} = \begin{cases} (x_2^{\mathcal{I}} - x_1^{\mathcal{I}}) \times (y_2^{\mathcal{I}} - y_1^{\mathcal{I}}) & \text{if} \quad x_2^{\mathcal{I}} > x_1^{\mathcal{I}}, y_2^{\mathcal{I}} > y_1^{\mathcal{I}} \\ 0 & \text{otherwise.} \end{cases}$$

5 Finding the coordinate of smallest enclosing box B^c:
$x_1^c = \min(\hat{x}_1^p, x_1^g)$, $\quad x_2^c = \max(\hat{x}_2^p, x_2^g)$,
$y_1^c = \min(\hat{y}_1^p, y_1^g)$, $\quad y_2^c = \max(\hat{y}_2^p, y_2^g)$.

6 Calculating area of B^c: $A^c = (x_2^c - x_1^c) \times (y_2^c - y_1^c)$.

7 $IoU = \frac{\mathcal{I}}{\mathcal{U}}$, where $\mathcal{U} = A^p + A^g - \mathcal{I}$.

8 $GIoU = IoU - \frac{A^c - \mathcal{U}}{A^c}$.

9 $\mathcal{L}_{IoU} = 1 - IoU$, $\quad \mathcal{L}_{GIoU} = 1 - GIoU$.

Experimental Results – YOLO v3

Table 1. Comparison between the performance of **YOLO v3** [21] trained using its own loss (MSE) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **test set of PASCAL VOC 2007.**

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
MSE [21]	.461	.451	.486	.467
$\mathcal{L}_{IoU}$	.466	.460	.504	.498
Relative improv %	1.08%	2.02%	3.70%	6.64%
$\mathcal{L}_{GIoU}$	**.477**	**.469**	**.513**	**.499**
Relative improv %	**3.45%**	**4.08%**	**5.56%**	**6.85%**

Table 2. Comparison between the performance of **YOLO v3** [21] trained using its own loss (MSE) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on 5K of the **2014 validation set of MS COCO**.

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
MSE [21]	.283	.312	.289	.330
$\mathcal{L}_{IoU}$	.292	.320	.312	.346
Relative improv %	3.18%	2.56%	7.96%	4.85%
$\mathcal{L}_{GIoU}$	**.301**	**.332**	**.325**	**.359**
Relative improv %	**6.36%**	**6.41%**	**12.46%**	**8.79%**

Table 3. Comparison between the performance of **YOLO v3** [21] trained using its own loss (MSE) as well as using $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **test set of MS COCO 2018**.

Loss / Evaluation	AP	AP75
MSE [21]	.311	.330
$\mathcal{L}_{IoU}$	.312	.338
Relative improv %	0.32%	2.37%
$\mathcal{L}_{GIoU}$	**.329**	**.356**
Relative improv %	**5.47%**	**7.30%**

Experimental Results – Faster R-CNN

Table 4. Comparison between the performance of **Faster R-CNN** [22] trained using its own loss (ℓ_1-smooth) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **test set of PASCAL VOC 2007**.

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
ℓ_1-smooth [22]	.370	.361	.358	.346
$\mathcal{L}_{IoU}$	.384	.375	.395	.382
Relative improv. %	3.78%	3.88%	10.34%	10.40%
$\mathcal{L}_{GIoU}$	**.392**	**.382**	**.404**	**.395**
Relative improv. %	**5.95%**	**5.82%**	**12.85%**	**14.16%**

Table 5. Comparison between the performance of **Faster R-CNN** [22] trained using its own loss (ℓ_1-smooth) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **validation set of MS COCO 2018.**

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
ℓ_1-smooth [22]	.360	.351	.390	.379
$\mathcal{L}_{IoU}$	.368	.358	.396	.385
Relative improv.%	2.22%	1.99%	1.54%	1.58%
$\mathcal{L}_{GIoU}$	**.369**	**.360**	**.398**	**.388**
Relative improv. %	**2.50%**	**2.56%**	**2.05%**	**2.37%**

Table 6. Comparison between the performance of **Faster R-CNN** [22] trained using its own loss (ℓ_1-smooth) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **test set of MS COCO 2018**.

Loss / Metric	AP	AP75
ℓ_1-smooth [22]	.364	.392
$\mathcal{L}_{IoU}$	**.373**	.403
Relative improv.%	**2.47%**	2.81%
$\mathcal{L}_{GIoU}$	**.373**	**.404**
Relative improv.%	**2.47%**	**3.06%**

Experimental Results – Mask R-CNN

Table 7. Comparison between the performance of **Mask R-CNN** [6] trained using its own loss (ℓ_1-smooth) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **validation set of MS COCO 2018**.

Loss / Evaluation	AP		AP75	
	IoU	**GIoU**	**IoU**	**GIoU**
ℓ_1-smooth [6]	.366	.356	.397	.385
$\mathcal{L}_{IoU}$	.374	.364	.404	.393
Relative improv.%	2.19%	2.25%	1.76%	2.08%
$\mathcal{L}_{GIoU}$	**.376**	**.366**	**.405**	**.395**
Relative improv. %	**2.73%**	**2.81%**	**2.02%**	**2.60%**

Table 8. Comparison between the performance of **Mask R-CNN** [6] trained using its own loss (ℓ_1-smooth) as well as $\mathcal{L}_{IoU}$ and $\mathcal{L}_{GIoU}$ losses. The results are reported on the **test set of MS COCO 2018**.

Loss / Metric	AP	AP75
ℓ_1-smooth [6]	.368	.399
$\mathcal{L}_{IoU}$	**.377**	.408
Relative improv.%	**2.45%**	2.26%
$\mathcal{L}_{GIoU}$	**.377**	**.409**
Relative improv.%	**2.45%**	**2.51%**

Extention