Towards Rotation Invariance in Object Detection Supplement

Agastya Kalra[1], Guy Stoppi[1], Bradley Brown[1], Rishav Agarwal[1] and Achuta Kadambi[1,2]
[1]Akasha Imaging, Palo Alto CA
[2]UCLA, Los Angeles CA
{agastya, guy.stoppi, bradley.brown, rishav}@akasha.im, {achuta}@ee.ucla.edu

In the supplement we have the following structure: Appendix A provides the algorithms for implementing our paper. Appendix B contains more detailed ablations for all the different axis-alignment methods we tried. Appendix C contains more results and information about shape optimization and the COCO shape. Appendix D has Faster-RCNN [3] results on COCO [2]. Appendix E contains more visualizations.

The code for Appendix C is also included in the supplement, as a PDF and a .ipynb file.

A. Algorithm Implementation

Here we include the implementation of both parts of our system. First the ellipse method in Algorithm 2 and then the loss function in Algorithm 1. It is clearly visible that both of these can be implemented in just a few lines of code.

Input: M, θ
M: Matched ground truth and predicted boxes
θ: Rotational uncertainty threshold
Output: L_{reg}: Regression loss
$l_{reg} \leftarrow 0$
for $(p, g_t \in M)$ **do**
  $l_{reg} = l_{reg} + l(p, g_t)$ — Standard L_{reg}
  if $IoU(p, g_t) < \max(0.5, C(\theta))$ **then** — RU Loss
    $l_{reg} = l_{reg} + l(p, g_t)$
  end
end
return l_{reg}

Algorithm 1: Implementation of RU Loss requires a simple one-line code change.

Input: B_i, θ
$B_i = [x, y, h, w]$: Input bounding box
(x, y): Center of the box
(h, w): Size of the box
θ: Angle of rotation data aug
s: scale of octagon (if used)
Output: $B_o = [x', y', h', w']$:Output bounding box

$h_2, w_2 \leftarrow h \,/\, 2, w \,/\, 2$

$X \leftarrow [w_2, w_2, -w_2, -w_2]$ — Largest Box
$Y \leftarrow [h_2, -h_2 - h_2, h_2]$

$X \leftarrow [w_2 sin(1^o), w_2 sin(2^o) ... w_2 sin(360^o)]$ — Ellipse
$Y \leftarrow [h_2 cos(1^o), h_2 cos(2^o) ... h_2 cos(360^o)]$

for $x_i \in X$ **do**
  $x_i \leftarrow x_i + x$
end
for $y_i \in Y$ **do**
  $y_i \leftarrow y_i + y$
end
$X_r, Y_r \leftarrow rotate(X, Y, \theta)$
$x_{min}, x_{max}, y_{min}, y_{max} =$
$min(X_r), max(X_r), min(Y_r), max(Y_r)$
$w', h' \leftarrow x_{max} - x_{min}, y_{max} - y_{min}$
$x', y' \leftarrow (x_{max} + x_{min}) \,/\, 2, (y_{max} + y_{min}) \,/\, 2$
$B_o \leftarrow x', y', w', h'$
return B_o

Algorithm 2: The algorithm for rotating a bounding box label

B. Our Alternative Methods

In this section we present a more detailed description of the other methods mentioned in the paper and show two methods that are competitive with the ellipse in performance.

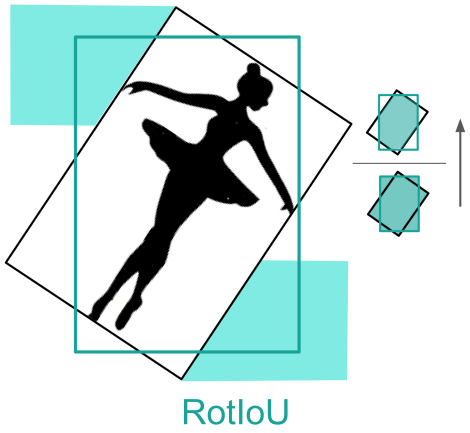
Each method is accompanied by a mathematical definition, using terms defined in the original paper. For instance, $\mathcal{R}^\theta$ is the rotation operator and $\mathcal{B}$ converts a shape to its axis-aligned bounding box.

$Q^\theta_{b_0}$ is the set of all new possible labels for a given original label b_0 after rotating the annotation by θ degrees. Finally $\hat{b}^\theta$ for a given method represents the method's estimate for the new label after rotating the original label by θ degrees.

B.1. RotIoU

The "RotIoU" idea was to select the box that maximizes IoU with the rotated original label. We constrain the optimization to the valid set of boxes, Q^θ_b, shown as the teal regions in Figure 1.

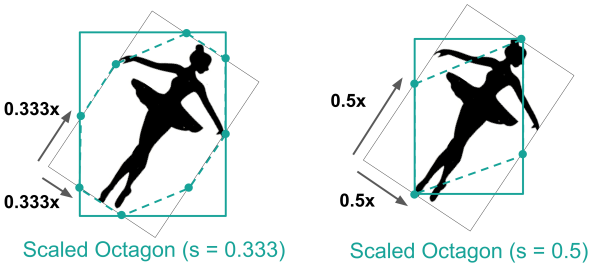
While RotIoU had fairly good performance, it is simply a worse version of Expected IoU. Conceptually RotIoU maximizes IoU with the rotated original label, whereas Expected IoU maximizes IoU with expected ground truth labels. As shown in the main paper, maximizing Expected IoU with the ellipse leads to better performance.



$$\hat{b}^\theta_{RotIoU} = \underset{b^\theta \in Q^\theta_{b_0}}{\operatorname{argmax}} IoU(b^\theta, \mathcal{R}^\theta(S_{largest})) \quad (1)$$

Figure 1: The RotIoU method aims to make the new label ($\hat{b}^\theta_{RotIoU}$) as similar as possible to the old label ($R^\theta(S_{largest})$).

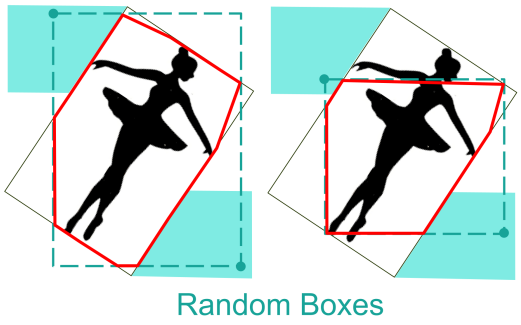
B.2. Scaled Octagon



$$\hat{b}^\theta{}_{oct}(s) = \mathcal{B}(\mathcal{R}^\theta(\mathcal{S}_{Oct}(s))) \quad (2)$$

Figure 2: The Scaled Octagon method interpolates between the Largest Box ($s = 0$) and a Diamond ($s = 0.5$).

Our first instinct was to simply introduce a scaling factor on largest box method, however this wouldn't guarantee a valid box. We then created the Scaled Octagon. We simply interpolate between a diamond and the largest box with a scaling factor s. As shown in Figure 2, we can see that $s = 0.5$ gives a diamond and $s = 0.0$ gives the largest box. The best value of s empirically is $s = 0.333$. We found this



$$\hat{b^{\theta}}_{random} = b_i \text{ for some } b_i \in Q^{\theta}_{b_0} \tag{3}$$

Figure 3: Visualization of the Random Boxes method. The red outlines show random shapes that were used for the axis-alignment.

both maximizes EIoU and achieves a shape very close to the Ellipse. We didn't include this in the main paper, since the final shape and EIoU are very similar to the ellipse. We include results for this in the Best results section.

Formally, "Scaled Octagon s" is defined as below. We define a shape by an ordered list of its (x,y) coordinates.

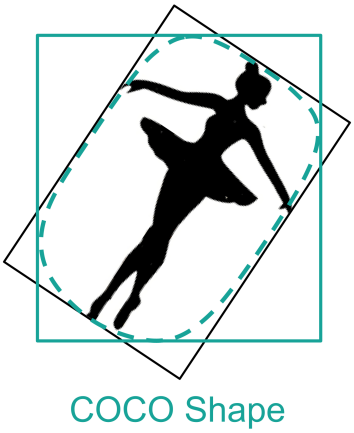
$$\begin{aligned}
&\text{Original Label} = \{ \\
&\quad (x_1, y_1), (x_1, y_2), (x_2, y_2), (x_1, y_2) \\
&\} \\
&x_1' = sx_1 + (1-s)x_2 \\
&x_2' = (1-s)x_1 + sx_2 \\
&y_1' = sy_1 + (1-s)y_2 \\
&y_2' = (1-s)y_1 + sy_2 \\
&\text{Oct}_s = \{ \\
&\quad (x_1',\ y_1),\ (x_2',\ y_1),\ (x_2,\ y_1'),\ (x_2,\ y_2') \\
&\quad (x_2',\ y_2),\ (x_1',\ y_2),\ (x_1,\ y_2'),\ (x_1,\ y_1') \\
&\}
\end{aligned}$$

B.3. Random Boxes

The Random Boxes method consists of selecting a random shape and then using that shape for axis-alignment of the new label.

This idea was more to form a baseline for our other methods. Many of our methods are focused on selecting an optimal shape for axis-alignment. We use Random Boxes's results to ensure that our selection of the optimal shape is indeed better than average. It is interesting to note that the random shape baseline outperforms the largest box significantly, showing the importance of replacing the Largest Box.

B.4. COCO Shape



$$\hat{b^{\theta}}_{COCO} = \mathcal{B}(\mathcal{R}^{\theta}(S_{COCO})) \tag{4}$$

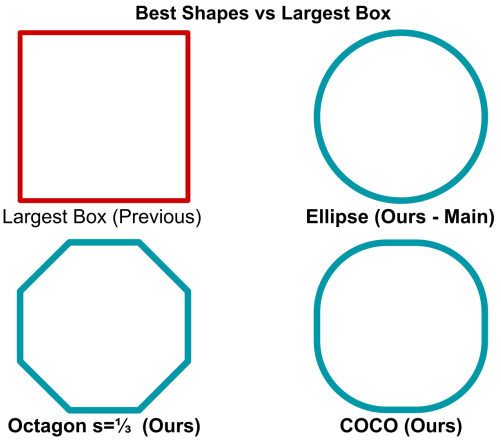
Figure 4: Visualization of the COCO shape method. The green dashed outline is the estimated COCO shape for this aspect ratio.

Instead of maximizing EIoU using a random shape distribution, we can optimize over shapes from the COCO dataset. In Appendix C Figure 7, we show plots of finding the optimal shape for Expected IoU over the COCO dataset. The resulting shape was not quite elliptical, but far more elliptical than Largest Box, as can be seen in Figure 4.

B.5. Our Best Methods

The results in Figure 6 show our best methods. We show that they all have similar EIoUs, and similar performance on their own. However we find that when combined with RU Loss, the Ellipse is clearly better. We do not know why exactly this occurs and can be an interesting direction of future research. This paper focuses on replacing the Largest Box with a robust alternative. We can see from the table above this is true.

While it is interesting to understand the nuances of different methods, we focus the main paper on proposing the best performing alternative to the Largest Box method - Ellipse + RU Loss.



	EIoU_{COCO}	EIoU_{Rand}	mAP			
Angles	$[0° - 360°]$	$[0° - 360°]$	**0°**	**10°**	**20°**	**30°**
Previous Shape						
Largest Box	69.4%	60.7%	35.20	28.37	22.34	18.47
Our Best Shapes						
COCO	**78.4%**	71.7%	37.51	36.37	33.47	29.96
Octagon $s = 0.333$	78.1%	72.7%	37.54	36.62	**33.89**	**30.42**
Ellipse	77.8%	**72.9%**	**38.21**	**36.83**	33.59	29.95
+ Our RU Loss						
COCO	**78.4%**	71.7%	38.51	37.85	35.69	32.53
Octagon 0.333	78.1%	72.7%	38.59	37.84	35.73	32.36
Ellipse	77.8%	**72.9%**	**39.33**	**38.31**	**36.00**	**32.72**

Figure 6: **Comparison of the best methods with and without RU Loss**. We find that the methods are quite competitive without RU Loss, but with RU Loss the ellipse is clearly better. We also note that all three methods have similar Expected IoU on COCO and Random Shapes.

C. Expected IoU Optimization

In this section we first provide additional details on exactly how the optimization was done. We also show that the starting shape for the optimization does not affect the final converged shape. Finally we show that if we use the COCO dataset shapes instead of random shapes it converges very close to an ellipse. We do not use this shape in our training since it is obtained from segmentation labels.

C.1. Optimization Details

In this section we walk through a high level on how each part of this optimization is done. We also provide a working python jupyter notebook in the supplement as both a pdf and as an ipynb. We recommend the reader to examine that for all the details.

We revisit equation 15 from the original paper:

$$\widehat{S} = \underset{S \in \mathcal{V}_{\mathbf{b^0}}}{\text{argmax}} \sum_{\theta} \frac{1}{K} \sum_{k=1}^{K} \left[IoU(\mathcal{B}(\mathcal{R}^{\theta}(S)), \mathbf{b}_k^{\theta}\right]. \quad (5)$$

The goal is to find the optimal shape that, when used for rotation augmentation, will lead to the highest expected IoU with potential ground truth boxes. To summarize from the paper: $\widehat{S}$ is the optimal shape, $\mathbf{b^0}$ is initial bounding box, θ is the angle of rotation, $\mathcal{B}$ is the operator to convert a shape to a bounding box by taking the min/max x, y coordinates of the shape, $\mathcal{V}_{\mathbf{b^0}}$ is the set of all possible shapes that would lead to $\mathbf{b^0}$, $\mathcal{R}^{\theta}$ is a rotation operator of θ degrees, and b_k^{θ} is a sampled ground truth box from a dataset $\mathcal{P}$. More details can be found in the paper.

This equation has three main components for implementation: the sampling method for b_k^{θ}, evaluating the cost function, and performing the gradient ascent optimization. All of these are implemented in the code files provided in the supplement zip.

C.1.1 Sampling b_k^{θ}

To sample a random ground truth box, we first randomly sample a shape S_k from the dataset that could have led to the initial bounding box $\mathbf{b^0}$. We do this for both a random dataset of shapes, which we use in the paper, and COCO. We then rotate this shape by θ and use the operator $\mathcal{B}$ to get a potential bounding box: $b_k^{\theta} = \mathcal{B}(\mathcal{R}^{\theta}(S_k))$

Random Dataset: This dataset is the one described in the paper: given a bounding box b, random shapes (i.e. V_b) are generated by selecting one point on each of the four walls of the bounding box and connecting them together into a polygon. This way we guarentee that taking the minimum and maximum points of the shape will lead to $\mathbf{b^0}$. This method does not use the segmentation labels of any dataset, and therefore is applicable to object detection.

MS CoCo Dataset: We can sample shapes for the MS CoCo dataset by iterating through the MS CoCo dataset, selecting a shape S_k such that $\mathcal{B}(S_k)$ is fairly close to the bounding box $\mathbf{b^0}$'s aspect ratio, and then stretching it so it fits completely within $\mathbf{b^0}$. This method leads to the optimal box on MS CoCo, but uses segmentation labels hence we do not use this to train. However this sampling still leads to a shape that achieves an expected IoU close to that of an ellipse.

C.1.2 Evaluating the Cost Function

To evaluate the cost function, we randomly sample $K = 1000$ boxes for each 5 degree interval of θ. We then evaluate the cost function given an input shape S by averaging IoU across all K boxes for each interval of θ.

C.1.3 Gradient Ascent

Performing gradient ascent directly on the cost function leads to poor convergence and many local minimums. We adapt gradient ascent using the following observation: to perform a rotation augmentation, we do not need the full shape $\widehat{S}$, we only need the convex hull $\mathcal{H}$ of the outline $\mathcal{O}$

of the shape $\mathcal{H}(\mathcal{O}(\widehat{S}))$. We also define a shape by $\mathcal{U}$, uniformly sampling, x, y coordinates along the boundary of the convex hull of the shape. So a single step of gradient ascent works as follows:

1. Compute the gradient G of the cost function w.r.t. the current shape S.
2. Update S by adding G times some learning rate α.
3. Update S by sampling 200 random points along the boundary of the convex hull of $S = \{x, y | x, y \sim \mathcal{U}(\mathcal{O}(\mathcal{H}(S)))\}$.

The last step is key to achieving proper convergence.

C.2. Examples

The graphs in this section show the optimization process being applied to different bounding boxes on the datasets above, as well as with multiple starting shapes. We see the convergence doesn't change depending on starting shape, and in both cases the expected IoU of the ellipse (dotted blue) is signficantly higher than that of the previous largest box (dotted red).

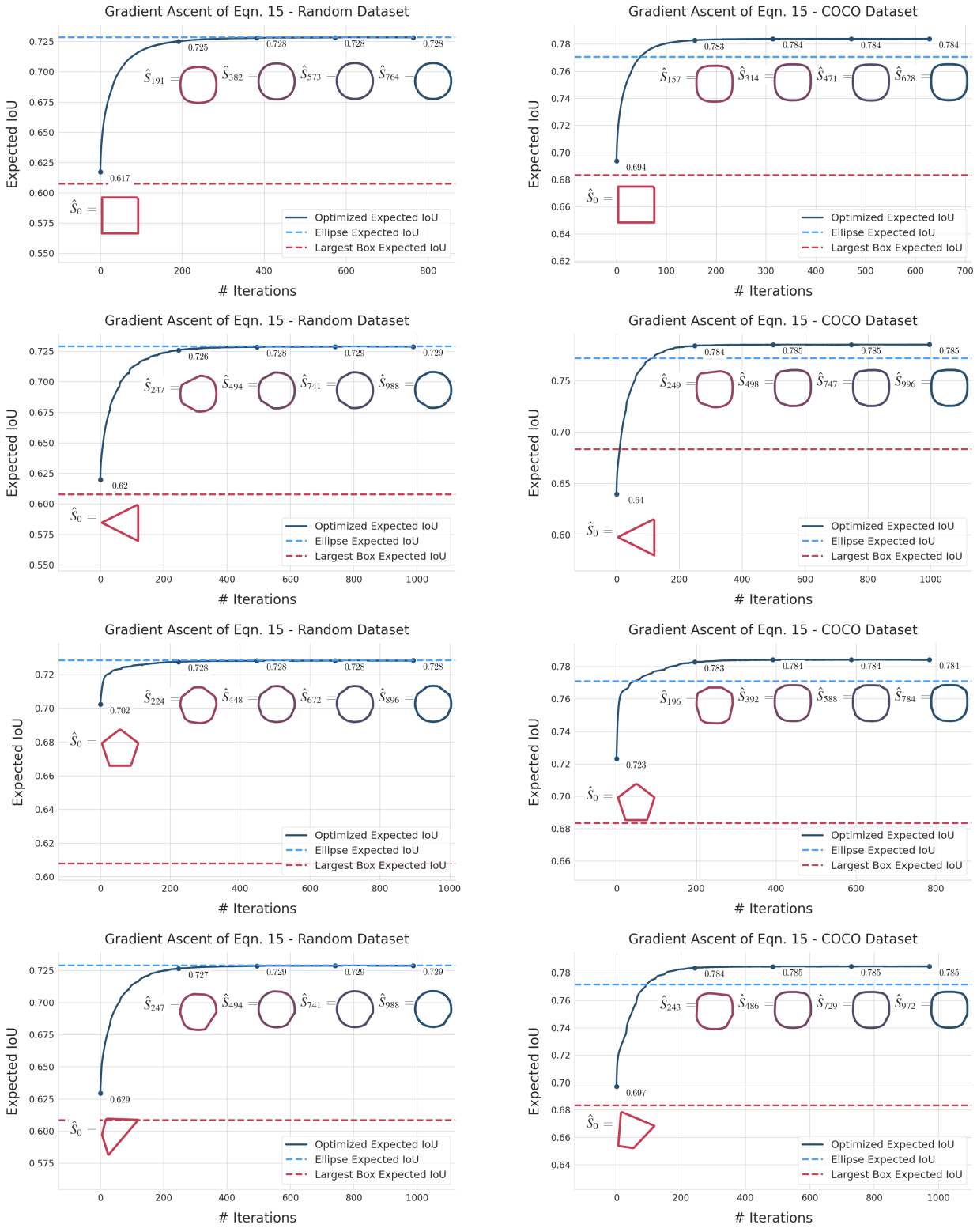


Figure 7: Finding the optimal shape to use when rotating a box (with aspect ratio 1:1) in order to maximize Eqn. 15, starting from the square, triangle, pentagon, and a random shape. **The final optimal shape is much more rounded than the largest box method.** In addition, the expected IoU of using an Ellipse is always much higher than the expected IoU of using the Largest Box.

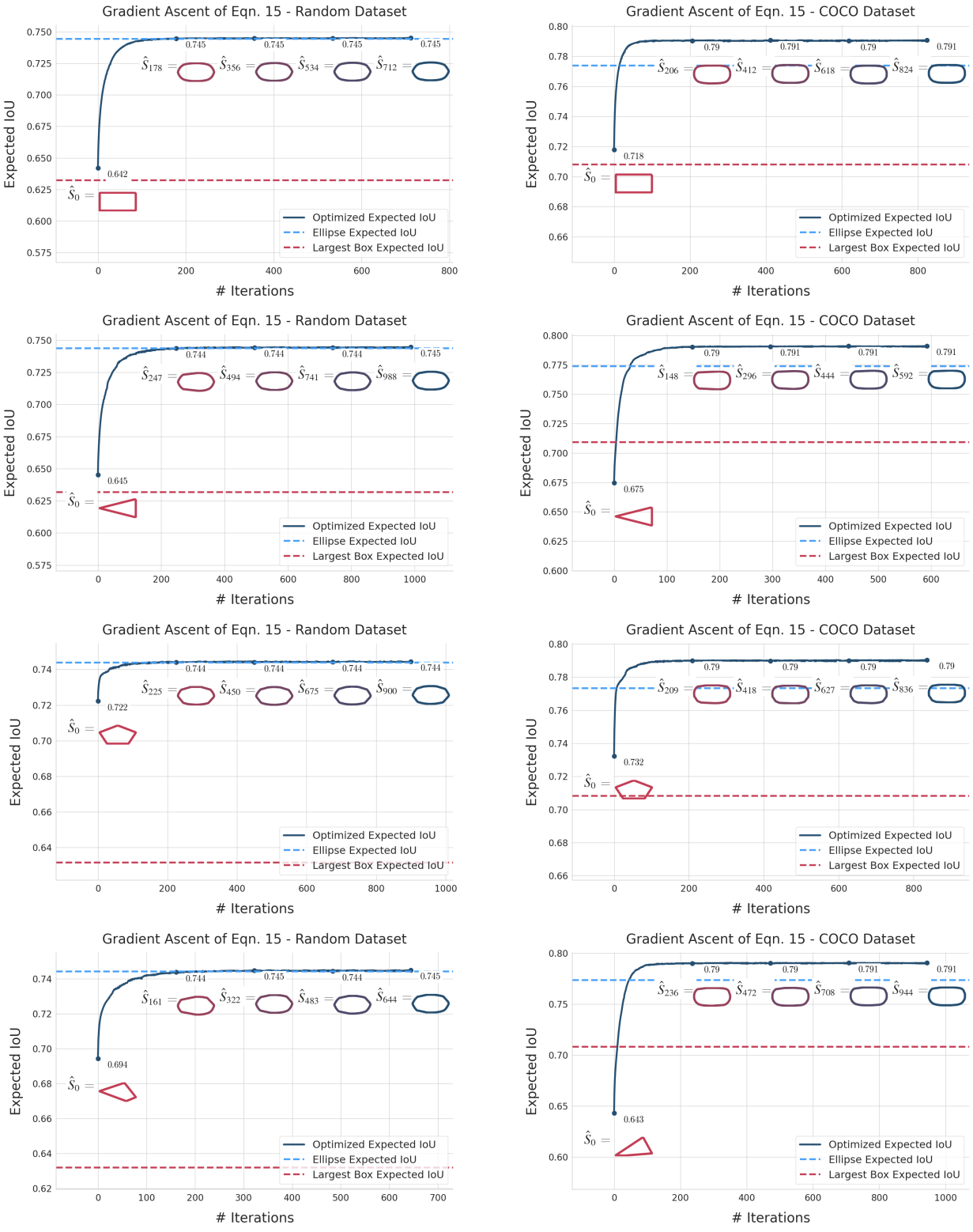


Figure 8: Finding the optimal shape to use when rotating a box (with aspect ratio 1:2) in order to maximize Eqn. 15, starting from the square, triangle, pentagon, and a random shape. **The final optimal shape is much more rounded than the largest box method.** In addition, the expected IoU of using an Ellipse is always much higher than the expected IoU of using the Largest Box.

D. Faster R-CNN

We test our method with Faster R-CNN on the COCO validation dataset (Table 1). The results look very similar to the RetinaNet results from the main paper, showing that our method generalizes across both one-stage and two-stage architectures.

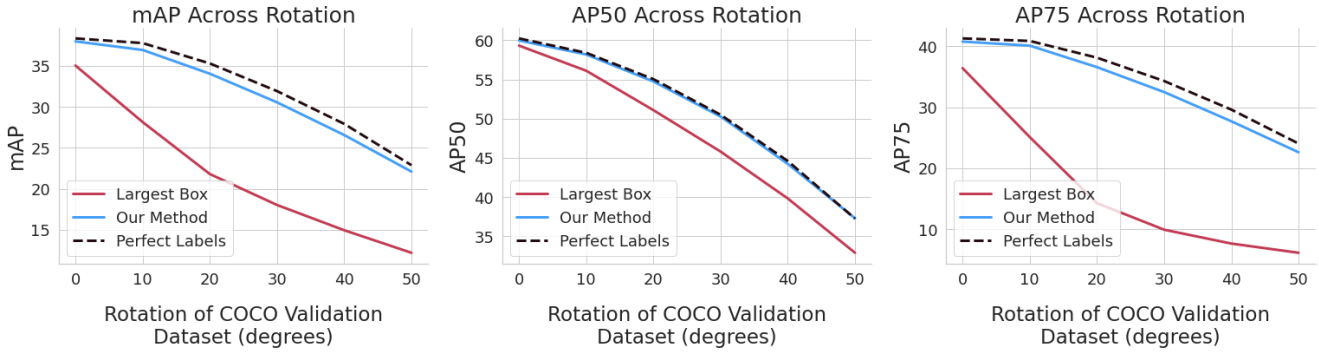


Figure: **The above results for Faster-RCNN are very similar to RetinaNet results from the main paper.** Our method closes the rotation gap across all three metrics.

	mAP				**AP50**				**AP75**			
	0°	**10°**	**20°**	**30°**	**0°**	**10°**	**20°**	**30°**	**0°**	**10°**	**20°**	**30°**
No Rotation	38.06	35.88	31.49	26.71	59.64	56.06	49.87	43.13	40.84	38.86	33.94	28.56
Largest Box	35.07	28.16	21.81	18.02	59.31	56.11	51.09	45.80	36.42	25.12	14.29	9.94
relative improvement	-7.85%	-21.53%	-30.76%	-32.52%	-0.55%	0.08%	2.44%	6.20%	-10.81%	-35.36%	-57.89%	-65.20%
Our Method	**38.02**	**36.97**	**34.07**	**30.57**	**59.96**	**58.18**	**54.74**	**50.27**	**40.77**	**40.10**	**36.58**	**32.49**
relative improvement	**-0.09%**	**3.04%**	**8.18%**	**14.46%**	**0.54%**	**3.77%**	**9.77%**	**16.55%**	**-0.17%**	**3.19%**	**7.77%**	**13.78%**
Perfect Labels	38.39	37.81	35.33	31.94	60.23	58.39	55.04	50.50	41.29	40.87	38.13	34.30
relative improvement	0.86%	5.38%	12.18%	19.60%	0.99%	4.15%	10.36%	17.08%	1.11%	5.17%	12.34%	20.13%

Table 1: For Faster-RCNN, our method of Ellipse + RU Loss performs close to perfect labels across all metrics at generalizing to new rotations on COCO val2017. This is similar to the results for RetinaNet [1] from the main paper.

E. Sample Images

In this section we present sample predictions from COCO in Figures 9 and 10. We give sample predictions from Pascal VOC in Figure 11.

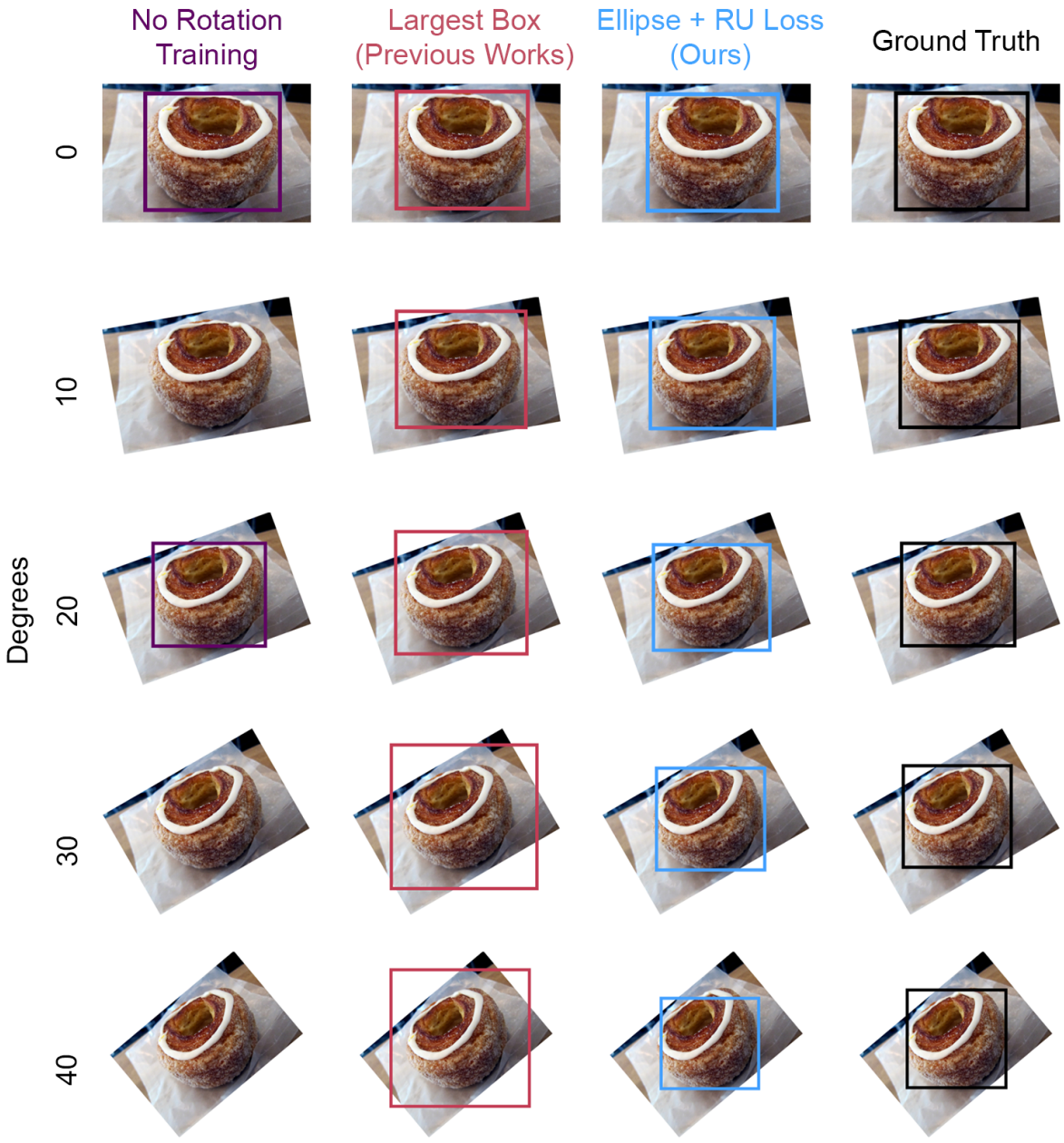


Figure 9: **Predictions on a single image over several positive rotations. As the rotation angle increases, training without rotation leads to missed objects and the largest box predicts oversized boxes.**

Figure 10: **Predictions on a single image over several negative rotations.**

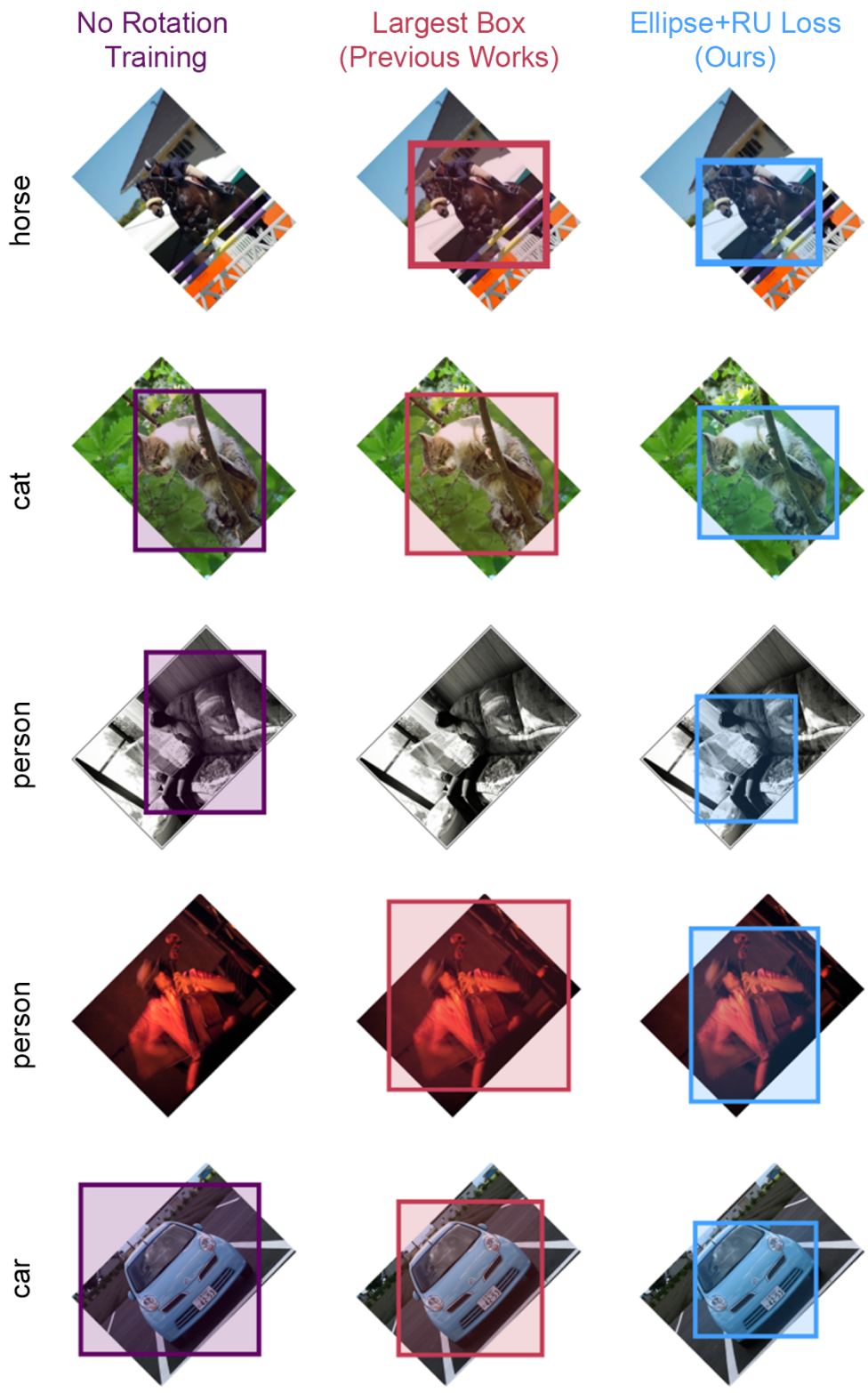


Figure 11: **Predictions on the Pascal VOC dataset. Similarly to COCO, training without rotation leads to missed objects and training with largest box leads to oversized predictions.**

References

[1] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 9

[2] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1