Abstract

This paper presents Mask-aware Intersection-over-Union (maIoU) for assigning anchor boxes as positives and negatives during training of instance segmentation methods. Unlike conventional IoU or its variants, which only considers the proximity of two boxes; maIoU consistently measures the proximity of an anchor box with not only a ground truth box but also its associated ground truth mask. Thus, additionally considering the mask, which, in fact, represents the shape of the object, maIoU enables a more accurate supervision during training. We present the effectiveness of maIoU on a state-of-the-art (SOTA) assigner, ATSS, by replacing IoU operation by our maIoU and training YOLACT, a SOTA real-time instance segmentation method. Using ATSS with maIoU consistently outperforms (i) ATSS with IoU by $\sim 1$ mask AP, (ii) baseline YOLACT with fixed IoU threshold assigner by $\sim 2$ mask AP over different image sizes and (iii) decreases the inference time by 25% owing to using less anchors. Then, exploiting this efficiency, we devise maYOLACT, a faster and $+6$ AP more accurate detector than YOLACT. Our best model achieves 37.7 mask AP at 25 fps on COCO test-dev establishing a new state-of-the-art for real-time instance segmentation. Code is available at https://github.com/kemaloksuz/Mask-aware-IoU.

1 Introduction

Instance segmentation is a visual detection problem which aims to classify and locate each object in an image by pixel-level masks. To be able to handle objects of different numbers, locations and scales; SOTA methods [3, 14, 35] employ a dense set of object hypotheses, generally represented by boxes or points, and ensure a maximum coverage of the objects. This coverage necessitates a large number of object hypotheses ($\sim 20k$ per image in YOLACT [3] for images of size $550 \times 550$) to be assigned to ground truths boxes; generally known as the assignment problem [17, 42].

This assignment problem is commonly tackled by employing heuristic rules. One common rule to assign object hypotheses represented by boxes (i.e. anchors) is using a “fixed IoU threshold” [1, 11, 14], in which an anchor, $\hat{B}$, can be assigned with a ground truth (i.e positive), $B$, when their Intersection-over-Union (IoU), defined as $\text{IoU}(\hat{B}, B) = |\hat{B} \cap B|/|\hat{B} \cup B|$, exceeds a pre-determined threshold, $\tau$. If anchor $\hat{B}$ cannot be assigned to any ground-truth
Anchors with low IoU but including significant object parts

Anchors with high IoU but missing significant object parts

Figure 1: Sample cases illustrating the need for mask-aware IoU (maIoU). Green boxes denote ground truth, red boxes are real anchors produced during the training. Left panel shows cases where the anchor covers a significant part of the object pixels but IoU is low (i.e. less than positive threshold of 0.50 for YOLACT). maIoU is higher than IoU for these cases, potentially correcting the assignment. Right panel shows cases where the anchor covers only a small part of the object pixels but IoU is high (so, anchors are positive). maIoU is lower than IoU, potentially correcting the assignment. Images are from COCO [19].

B (i.e. \( \text{IoU}(\hat{B}, B) < \tau, \forall B \)), then \( \hat{B} \) is assumed to be a background (i.e. negative) example. A different set of recent methods showed for object detection [17, 18, 42] that assigning the anchors using an “adaptive IoU threshold” determined for each ground truth improves the performance. Still, assignment methods heavily rely on IoU as the de facto proximity measure between ground truths and anchors.

Despite its popularity, IoU has a certain drawback: The IoU between an anchor box and a ground truth box solely depends on their areas, thereby ignoring the shape of the object, e.g. as provided by a segmentation mask. This may give rise to undesirable assignments due to counter-intuitively lower or higher IoU scores. For example, the IoU might be high, implying a positive anchor, but only a small part of the object is included in the anchor; or the IoU may be low, implying a negative anchor, but a large part of the object is included in the anchor. Fig. 1 presents examples for such cases, arising due to objects with unconventional poses, occlusion and objects with articulated or thin parts. We will show (in Section 3.1, Fig. 3) that such examples tend to produce larger loss values and adversely affect training.

In this paper, we introduce mask-aware IoU (maIoU), a novel IoU measure for anchor assignment in instance segmentation by exploiting the ground truth masks of the objects, normally used only for supervision. Specifically, unlike IoU, which equally weights all pixels, maIoU yields a proximity measure between 0 and 1 among an anchor box, a ground truth box and a ground truth mask by promoting the pixels on the masks, thereby providing a more consistent assignment (Fig. 1). Since a naive computation of maIoU is impractical, we present an efficient algorithm with training time similar to the baseline. YOLACT with maIoU-based ATSS assigner consistently improves ATSS assigner with IoU by \( \sim 1 \) mask AP and standard YOLACT (i.e. fixed IoU threshold) by \( \sim 2 \) mask AP, and also decreases inference time of YOLACT. Finally, utilizing this efficiency gap, we build maYOLACT detector reaching 37.7 mask AP at 25 fps and outperforming all real-time counterparts.
Table 1: IoU Variants, their inputs and primary purposes. IoU variants assign a proximity measure based on the properties (prop.) of two inputs (Input 1 and Input 2). In practice, existing variants compare the inputs wrt. the same properties (i.e. either boxes or masks). Our Mask-aware IoU (maIoU) can uniquely compare a box with a box and a mask. With this, maIoU compares anchors (i.e. only box) with ground truths (box and mask) in order to provide better anchor assignment. *: GIoU is also used as a performance measure.

<table>
<thead>
<tr>
<th>IoU Variant</th>
<th>Input 1 prop.</th>
<th>Input 2 prop.</th>
<th>Primary purpose as proposed in the paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask IoU [8, 19]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Boundary IoU [9]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Generalized IoU [29]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Distance IoU [43]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Complete IoU [44]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Mask-aware IoU (Ours)</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

2 Related Work

Deep Instance Segmentation. In general, deep instance segmentation methods have followed object detection literature. The pioneering Mask R-CNN model [14] and its variations [15, 26] extended Faster R-CNN [28] by incorporating a mask prediction branch into the two-stage detection pipeline. Similarly, anchor-based one-stage methods were also adapted for instance segmentation by using an additional mask prediction branch, e.g. YOLACT [3] and YOLACT++ [4] employed a YOLO-like architecture; and PolarMask [39] and PolarMask++ [40], both anchor-free methods, adapted FCOS [30] for instance segmentation. Differently, SOLO variants [34, 35] classify the pixels based on location and size of each instance.

Anchor Assignment in Instance Segmentation. Deep instance segmentation methods using anchors as object hypotheses label anchors based on “fixed IoU threshold” assignment rule: The anchors with IoU larger than τ⁺ with a ground truth box are assigned as positive; while the anchors whose maximum IoU with ground truths is less than τ⁻ are assigned as negatives; and the remaining anchors whose maximum IoU is between τ⁻ and τ⁺ are simply ignored during training. To illustrate, YOLACT variants [3, 4] and RetinaMask [23] use τ⁻ = 0.40 and τ⁺ = 0.50; while the first stage of Mask R-CNN (i.e. region proposal network [28]) sets τ⁻ = 0.30 and τ⁺ = 0.70; and finally its second stage [29] uses τ⁻ = τ⁺ = 0.50.

Adaptive Anchor Assignment Methods in Object Detection. Recently, adaptive anchor assignment strategies are shown to perform better than fixed IoU threshold in object detection: ATSS [42] uses top-k anchors wrt. IoU to determine an adaptive IoU threshold for each ground truth (Section 3.4 provides more details on ATSS.) and PAA [17] computes a score of each anchor including Generalized IoU and fits the distribution of these scores to a two-dimensional Gaussian Mixture Model to split positives and negatives for each ground truth. Similarly, Ke et al. [16] and Li et al. [18] identify positives and negatives by using different scoring functions of the predictions. However, these methods are devised and tested for object detection, and thus, do not utilize object masks.

Other IoU Variants. Over the years, many IoU variants have been proposed – see Table 1 for a comparative summary. One of the most related IoU variants is Mask IoU, which is used to measure the detection mask quality with respect to (wrt.) the ground truth mask during evaluation [8, 19]. Similarly, Boundary IoU [9] evaluates detection masks by giving higher weights to the pixels closer to the boundaries. Note that these IoU variants compare
only two masks, and unlike our maIoU, they cannot compare a box with another box and its associated mask. The other IoU variants are all devised to measure the proximity of two boxes: Generalized IoU (GIoU) \cite{giou} uses the minimum enclosing box in order to measure the proximity of boxes when boxes do not intersect (i.e. their IoU is 0); Distance IoU \cite{distance} adds a penalty parameter based on the minimum enclosing box and the distance between the centers of boxes; Complete IoU \cite{complete} additionally considers aspect ratio differences of the boxes. These IoU-variants compute the overlap at the box level and neglect object shape; and also, they are mainly used as loss functions, not as a positive-negative assignment criterion. 

**Comparative Summary.** By measuring the proximity of an anchor box with a ground truth, our maIoU is designed for anchor-based models, which have been using a fixed IoU threshold for assigning anchors as the dominant approach, thereby ignoring object shape. We first show that ATSS \cite{atss}, an adaptive anchor assigner, yields better performance on YOLACT \cite{yolact}. Then, we propose maIoU, as the first IoU variant that can measure the proximity of an anchor box with a ground truth box and ground truth mask (Table 1). Replacing IoU of ATSS by our maIoU improves the performance of this strong baseline. We also investigate GIoU and DIoU for anchor assignment. Since they rely only on boxes, our maIoU provides more discriminative information then these IoU variants. Finally, besides our maIoU, adopting recently proposed improvements into YOLACT detector, we build maYOLACT detector, which outperforms its counterparts while being more efficient as well (Section 4).

## 3 Methodology

This section first presents an analysis on fixed-threshold IoU assigner in Section 3.1. Then, Section 3.2 defines maIoU and Section 3.3 provides an efficient algorithm to compute maIoU. Finally, Section 3.4 incorporates our maIoU into the SOTA ATSS assigner \cite{atss} to label anchors during the training of instance segmentation methods.

### 3.1 The Mask-over-box Ratio

Our analysis is based on an intuitive measure to represent the rate of the ground truth mask in a box, defined as the Mask-over-box (MOB) ratio as follows.
Definition 1. Mask-over-box (MOB) ratio of a box (i.e. anchor or ground truth), \( \hat{B} \), on a
ground truth mask, \( M \), is the ratio of (i) the area of the intersection of the mask and the box,
and (ii) the area of \( \hat{B} \) itself:

\[
\text{MOB}(\hat{B}, M) = \frac{|\hat{B} \cap M|}{|\hat{B}|}.
\]

\( \text{MOB}(\hat{B}, M) \in [0, 1] \) and less object pixels from \( M \) in \( \hat{B} \) implies a lower MOB ratio. When
\( \hat{B} \) is the ground truth box of \( M \) (i.e. \( \hat{B} = B \)), \(|M \cap B| = |M|\), and thus \( \text{MOB}(B, M) = |M|/|B| \).

We now analyse how losses on a trained standard YOLACT model (with ResNet-50
backbone), a SOTA instance segmentation method, and the number of anchors change wrt.
MOB ratio (Fig. 2) and make the following crucial observations:

(1) The loss value (i.e. error) of a trained YOLACT for an anchor is related to the
amount of mask pixels covered by that anchor (i.e. MOB ratio), which is ignored by
the standard fixed IoU threshold assigner. Fig. 2(a,b) present that the average and standard
deviation of the loss values of anchors that are close to the IoU assignment threshold (i.e.
the anchors with IoU between \([0.30 - 0.50]\) for negatives and \([0.50 - 0.70]\) for positives;
hence, anchors with similar IoUs) increase in all tasks (for negatives, it is only classification;
for positives, we look at all tasks – i.e. classification, box regression and segmentation)
as MOB ratio decreases/increases (i.e. implying covering less/more on-mask pixels) for
positive/negative anchors. Also, the numbers of anchors with larger losses are in the order
of thousands in all cases (Fig. 2(c)).

(2) Similar to anchors, MOB ratios of the ground truth boxes also vary significantly.
We observe in Fig. 3(a) that there exists a significant amount of ground truth boxes with low
MOB ratios (i.e. for 30 % of the ground truths, MOB ratio is less than 0.50).

Our observations suggest that IoU does not discriminate anchors with more on-mask
pixels from those with less on-mask pixels, which appears naturally due to varying MOB
ratios of the ground truths; and thus using IoU for the assignment of the anchors may not be
the best method.

3.2 Mask-aware Intersection-over-Union (maIoU)

Intuition. The main intuition behind mask-aware IoU (maIoU) is to reweigh the pixels
within the ground truth box, \( B \), such that on-mask pixels are promoted (in a way, the con-
tribution of off-mask pixels are reduced) by preserving the total energy of \( B \) (i.e. \(|B|\)).
We simply achieve this by distributing the contributions of the off-mask pixels uniformly over
the on-mask pixels in \( B \). Finally, maIoU is computed as an intersection-over-union between
\( \hat{B} \) (i.e. anchor box) and \( B \) with the new pixel weights in \( B \) (Fig. 3(b)).

Derivation. To facilitate derivation, we first reformulate intersection \( I \) between \( B \) and \( \hat{B} \)
in a weighted form as follows (\( w_m, w_{\bar{m}} \): the contributions of on-mask and off-mask pixels
respectively):

\[
I(B, \hat{B}) = \sum_{i \in B \cap \hat{B}} w = \sum_{i \in B \cap M} w_m + \sum_{i \in (\hat{B} \cap B - \hat{B} \cap M)} w_{\bar{m}},
\]

which effectively does not make use of the mask \( M \) since \( w = w_m = w_{\bar{m}} = 1 \) for IoU.

In maIoU, we discard the contribution of an off-mask pixel: \( w_{\bar{m}} = 0 \), and in order to
preserve the total energy, the reduced contribution from all off-mask pixels, which equals
\(|B| - |M|\), is distributed to the on-mask pixels uniformly. This will increase \( w_m \) by \((|B| - |M|)/|M|\):
\( w_m = 1 + ((|B| - |M|)/|M|) = 1 + |B|/|M| - |M|/|M| = |B|/|M| = 1/\text{MOB}(B, M) \).
Figure 3: (a) The distribution of MOB ratios of the ground truths on COCO training set. (b) How IoU and maIoU weights the pixels in ground truth box (see Fig. 1 top-left example for the image of this example). While IoU does not differentiate among on-mask ($w_m$) and off-mask ($w_m$) pixels, our maIoU sets $w_m = 0$ and weights each on-mask pixel by inverse MOB ratio considering object mask $M$. (c) Anchor count distribution (in log-scale) of IoU vs maIoU. While IoU & high-maIoU positively correlate, there are quite a few examples with low-IoU & high-maIoU and vice versa.

With these weights, the mask-aware intersection, $maI$ is defined by extending Eq. 2:

$$maI(\hat{B}, B, M) = \sum_{i \in \hat{B}\cap M} w_m + \sum_{i \notin (\hat{B}\cap B \cap M)} w_m = 0 w_m|\hat{B} \cap M| = \frac{1}{MOB(B, M)}|\hat{B} \cap M|. \quad (3)$$

Extending the definition of union (i.e. $|B| + |\hat{B}| - |B \cap \hat{B}|$) with this intersection concept:

$$maU(\hat{B}, B, M) = |B| + (|\hat{B}| - |B \cap B| + maI(\hat{B}, B, M)) - maI(\hat{B}, B, M) = |\hat{B} \cup B|, \quad (4)$$

which is equal to the conventional union. This is not surprising since our formulation preserves ground truth area (i.e. $|B|$). With the updated definitions, mask-aware IoU is simply mask-aware intersection over mask-aware union:

$$maIoU(\hat{B}, B, M) = \frac{maI(\hat{B}, B, M)}{maU(\hat{B}, B, M)} = \frac{1}{MOB(B, M)} \frac{|\hat{B} \cap M|}{|\hat{B} \cup B|}. \quad (5)$$

which, in effect, is the ratio of covered on-mask pixels by the anchor ($|\hat{B} \cap M|$) in the union of boxes ($|\hat{B} \cup B|$), normalized by on-mask pixel density in $B$ (i.e. $MOB(B, M)$).

**Interpretation.** Similar to IoU, $maIoU(\hat{B}, B, M) \in [0, 1]$ and a larger value implies a better localisation not only the boxes but also the ground truth mask (Fig. 3(b)). We visualize the anchor count distribution (in log-scale) of YOLACT on COCO minival on the space spanned by IoU and maIoU (Fig. 3(c)): While IoU and maIoU are positively correlated, there are quite a number of examples with low-IoU & high-maIoU and vice versa, hence the assignment rules based on maIoU is quite different than those based on IoU.

### 3.3 Computation of maIoU

Compared to IoU, computing maIoU involves two additional terms (Eq. 5): (i) $|M|$, the total number of mask-pixels (since calculating $|B|$ for $MOB(B, M)=|M|/|B|$ is trivial), and (ii) $|\hat{B} \cap M|$, the number of mask-pixels in the intersection. While the masks are included...
in the datasets and computing these two quantities is straightforward; it is impractical to compute them naively (i.e. brute force, see Table 3) considering the large number of anchors covering the image. For this reason, we employ integral images [9] on the binary ground truth masks, \(M\). More specifically, for \(M\) covering an image of size \(m \times n\), we compute its integral image \(\Theta^M\), an \((m+1) \times (n+1)\) matrix that encodes the total number of mask pixels above and to the left of each pixel. Accordingly, denoting the \((i,j)\)th element of \(\Theta^M\) by \(\Theta_{i,j}^M\), the last element of \(\Theta^M\) stores \(|M|\), i.e. \(|M| = \Theta_{m+1,n+1}^M\). As for the second term, assuming \(\hat{B}\) is represented by a top-left point \((x_1, y_1)\) and a bottom-right point \((x_2, y_2)\) such that \(x_2 > x_1\) and \(y_2 > y_1\), \(|\hat{B} \cap M|\) involves only four look-up operations, i.e. \(|\hat{B} \cap M| = \Theta_{x_2+1,y_2+1}^M - \Theta_{x_2+1,y_1}^M - \Theta_{x_1,y_2+1}^M + \Theta_{x_1,y_1}^M\). The overall algorithm to compute maIoU(\(\hat{B}, B, M\)) is presented in Alg. 1.

Algorithm 1
The algorithm for efficiently calculating mask-Aware IoU.

1: procedure MaskAwareIoU(\(\hat{B}, B, M\))
2: Compute \(|B|, |\hat{B} \cup B|\) and \(\Theta^M\) as integral image of \(M\) such that \(\Theta_{i,j}^M\) is \((i, j)\)th element of \(\Theta^M\)
3: Set \(|M| = \Theta_{m+1,n+1}^M\) and \(|\hat{B} \cap M| = \Theta_{x_2+1,y_2+1}^M - \Theta_{x_2+1,y_1}^M - \Theta_{x_1,y_2+1}^M + \Theta_{x_1,y_1}^M\).
4: Compute \(\text{MOB}(B,M) = \frac{|M|}{|B|}\) for ground truth \(B\)
5: return maIoU(\(\hat{B}, B, M\)) (Eq. 5)
6: end procedure

3.4 Incorporating maIoU into ATSS Assigner

ATSS assigner [42] is a SOTA assignment method used for object detection, yielding better performance than a fixed-threshold IoU assigner and simplifying the anchor design by using a single anchor per pixel unlike its predecessors with up to nine anchors per pixel [21]. ATSS assigner comprises three steps: (i) selecting top-\(k\) (i.e. conventionally \(k = 9\)) anchors (\(\hat{B}\)) wrt. the distance of the centers between \(B\) and \(\hat{B}\) for each ground truth \(B\) on each FPN level as “candidates”, (ii) filtering out the candidates using an adaptive IoU threshold, computed based on the statistics of these candidates for each \(B\), and (iii) filtering out the candidates, whose centers lie out of \(B\). The surviving candidates after steps (ii) and (iii) are the positive examples and the remaining examples are the negatives. Using our maIoU with ATSS (or any IoU-based assigner) is straightforward: In step (ii), we just replace IoU-based adaptive thresholding by maIoU-based adaptive thresholding.

4 Experiments

Dataset. We train all models on the COCO trainval set [18] (115K images), test them on the COCO minival set (5k images) unless otherwise stated.

Performance Measures. We mainly report AP-based performance metrics: COCO-style AP (AP, in short), APs where true positives are validated from IoUs of 0.50 and 0.75 (AP\(_{50}\) and AP\(_{75}\)), and APs for small, medium and large objects (AP\(_S\), AP\(_M\) and AP\(_L\) respectively). Furthermore, we also exploit the recent optimal Localisation Recall Precision (oLRP) Error [24, 25]. While AP is a higher-is-better measure, oLRP is a lower-is-better metric.

Implementation Details. We conduct our experiments on YOLACT [8], an anchor-based real-time instance segmentation method, using the mmdetection framework [5]. Following
Table 2: Comparison of different assigners and IoU variants on YOLACT. Considering the shapes of the objects, our ATSS w. our maIoU consistently outperforms its counterparts.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Assigner</th>
<th>AP↑</th>
<th>AP50↑</th>
<th>AP75↑</th>
<th>AP8</th>
<th>AP10↑</th>
<th>AP15↑</th>
<th>AP20↑</th>
<th>AP25↑</th>
<th>AP30↑</th>
<th>AP35↑</th>
<th>AP40↑</th>
<th>oLRP↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>fixed IoU threshold</td>
<td>24.8</td>
<td>42.4</td>
<td>25.0</td>
<td>7.3</td>
<td>26.0</td>
<td>42.0</td>
<td>78.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. IoU</td>
<td>25.3</td>
<td>43.5</td>
<td>25.5</td>
<td>6.8</td>
<td>27.3</td>
<td>43.8</td>
<td>77.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. DIoU</td>
<td>25.4</td>
<td>43.6</td>
<td>25.2</td>
<td>7.2</td>
<td>27.1</td>
<td>43.4</td>
<td>77.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. GloU</td>
<td>25.1</td>
<td>42.7</td>
<td>25.3</td>
<td>7.0</td>
<td>26.8</td>
<td>41.8</td>
<td>78.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. maIoU (Ours)</td>
<td>26.1</td>
<td>44.3</td>
<td>26.3</td>
<td>7.2</td>
<td>28.0</td>
<td>44.3</td>
<td>77.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>fixed IoU threshold</td>
<td>28.5</td>
<td>47.9</td>
<td>29.4</td>
<td>11.7</td>
<td>31.8</td>
<td>43.0</td>
<td>75.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. IoU</td>
<td>29.3</td>
<td>49.2</td>
<td>30.2</td>
<td>11.1</td>
<td>33.0</td>
<td>44.5</td>
<td>74.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. DIoU</td>
<td>29.5</td>
<td>49.5</td>
<td>30.1</td>
<td>11.7</td>
<td>33.2</td>
<td>44.9</td>
<td>74.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. GloU</td>
<td>29.1</td>
<td>48.6</td>
<td>30.0</td>
<td>12.0</td>
<td>32.2</td>
<td>43.3</td>
<td>74.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. maIoU (Ours)</td>
<td>30.4</td>
<td>50.3</td>
<td>31.4</td>
<td>11.5</td>
<td>33.9</td>
<td>46.3</td>
<td>73.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>fixed IoU threshold</td>
<td>29.7</td>
<td>50.0</td>
<td>30.4</td>
<td>14.2</td>
<td>32.8</td>
<td>43.7</td>
<td>74.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. IoU</td>
<td>30.8</td>
<td>51.8</td>
<td>31.2</td>
<td>14.1</td>
<td>35.0</td>
<td>44.0</td>
<td>73.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. DIoU</td>
<td>30.9</td>
<td>51.9</td>
<td>31.7</td>
<td>14.0</td>
<td>35.4</td>
<td>44.0</td>
<td>73.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. GloU</td>
<td>30.1</td>
<td>50.7</td>
<td>31.0</td>
<td>14.0</td>
<td>33.8</td>
<td>43.1</td>
<td>74.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATSS w. maIoU (Ours)</td>
<td>31.8</td>
<td>52.8</td>
<td>32.8</td>
<td>14.7</td>
<td>35.6</td>
<td>45.7</td>
<td>72.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Zhang et al. [39], when we use ATSS assigner, we keep $k = 9$ (Section 3.4) and simplify the anchor configuration by placing a single anchor on each pixel with an aspect ratio of 1 : 1 and a base scale of 4 unless otherwise stated. Also, with ATSS (with IoU, DIoU [35], GloU [34] or our maIoU), we keep classification and box regression loss weights as they are (1.0 and 1.5 respectively), and increase mask prediction loss weight from 6.125 to 8. When we replace the assigner, note that it affects the examples in all branches (i.e. classification, box regression, semantic and instance mask predictions). We train all models with 32 images distributed on 4 GPUs (8 images/GPU). The remaining design choices of YOLACT [3] are kept. We adopt ResNet-50 [13] as the backbone and resize the images during training and inference to $S \times S$ where $S$ can be either 400, 550 or 700 following Bolya et al. [3].

4.1 Ablation Experiments

In this section, we demonstrate that our maIoU improves upon other assigners based on IoU variants consistently and our Alg. 1 makes computation of maIoU feasible during training.

Using ATSS with IoU. We first replace the fixed IoU threshold assigner of YOLACT by ATSS with (w.) IoU and have a stronger baseline for our maIoU: ATSS w. IoU improves fixed IoU assigner by 0.5 – 1.1 mask AP over different scales (Table 2).

Replacing IoU of ATSS with maIoU. Replacing IoU in ATSS by our maIoU (Table 2) (i) improves fixed IoU assigner by 1.3, 1.9 and 2.1 mask APs for 400, 550 and 700 scales respectively, (ii) outperforms ATSS w. IoU by $\sim 1.0$ mask AP in all scales, (iii) performs also better than other IoU variants (i.e. ATSS w. GloU and ATSS w. DIoU). We note that the contribution of maIoU (i) on models trained by images with larger scales (700 vs. 400 in Table 2) and (ii) on larger objects (AP1 vs. AP5) are more significant. This is intuitive since the shape of the object gets more precise as the object gets larger.

Computing maIoU Efficiently. Computing maIoU for every anchor-ground truth pair during training by brute force is infeasible, i.e. it would take $\sim 3$ months to train a single model with 41.89 sec/iteration (Table 3). Using Alg. 1, we reduce the average iteration time by
Table 3: Avg. iteration time ($t$) of assigners. While brute force maIoU computation is inefficient (Alg. 1 is $X$); our Alg. 1 decreases $t$ by $\approx 70\times$ and has similar $t$ with Fixed IoU Thr. and ATSS w. IoU.

<table>
<thead>
<tr>
<th>Assigner</th>
<th>Alg. 1</th>
<th>$t$ (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed IoU Thr.</td>
<td>N/A</td>
<td>0.51</td>
</tr>
<tr>
<td>ATSS w. IoU</td>
<td>N/A</td>
<td>0.57</td>
</tr>
<tr>
<td>ATSS w. maIoU</td>
<td>$X$</td>
<td>41.89</td>
</tr>
<tr>
<td>ATSS w. maIoU</td>
<td>✓</td>
<td>0.59</td>
</tr>
</tbody>
</table>

$\sim 70\times$ to 0.59 sec/iteration, which is similar to other standard assigners (Table 3).

4.2 maYOLACT Detector: Faster and Stronger

Thanks to using fewer number of anchors, YOLACT trained with our ATSS w. maIoU assigner (underlined in Table 4) is $\sim 25\%$ faster than baseline YOLACT (33 vs. 27 fps$^1$), pointing out the importance of anchor design for the efficiency of real-time systems as well$^2$. Exploiting this run-time gap; our aim in this section is to extend the standard YOLACT using our maIoU and the recent improvements in order to make it competitive with the recent methods also by keeping the resulting detector to process images in real-time$^3$. To achieve that, we use (i) carafe$^4$ as the upsampling operation of FPN$^5$, (ii) deformable convolutions$^6$ in the backbone, (ii) two anchors with base scales 4 and 8 on each pixel, and (iv) cosine annealing with an initial learning rate of 0.008 by replacing the step learning rate decay. Effect of these improvements are presented in Table 4 and the resulting detector with these improvements is coined as maYOLACT. Note that our maYOLACT-550 detector is still faster than baseline YOLACT-550 and improves it by $+6.3$ mask AP and $+7.2$ box AP reaching 34.8 mask AP and 37.9 box AP (Table 4).

4.3 Comparison with State-of-the-art (SOTA)

Table 5 compares our maYOLACT with state-of-the-art methods on COCO test-dev.

Comparison with YOLACT variants. Achieving 35.2 mask AP, our maYOLACT-550 outperforms all YOLACT variants including the ones with larger backbones (e.g. YOLACT-550++ with ResNet-101) and larger scales (e.g. YOLACT-700). Besides, different from YOLACT++$^7$, which is $\sim 25\%$ slower than YOLACT (see Table 6 in Bolya et al.$^8$), our maYOLACT-550 is faster than YOLACT-550 (Table 4), and still keep 7 mask AP gain also on COCO test-dev reaching 35.2 mask AP (Table 5).

Comparison with real-time methods. Without multi-scale training as in Solov2$^9$ or specially designed backbone as in CenterMask$^{10}$; our maYOLACT-700 reaches 37.7 mask AP at 25fps and outperforms existing real-time counterparts. Besides, our best model

<table>
<thead>
<tr>
<th>Method</th>
<th>AP</th>
<th>AP$^\text{box}$</th>
<th>fps</th>
<th>Anchor #</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLACT-550</td>
<td>28.5</td>
<td>30.7</td>
<td>27</td>
<td>$\sim 19.2K$</td>
</tr>
<tr>
<td>+ ATSS w. maIoU</td>
<td>30.4</td>
<td>32.5</td>
<td>33</td>
<td>$\approx 6.4K$</td>
</tr>
<tr>
<td>+ Carafe FPN$^4$</td>
<td>31.4</td>
<td>33.3</td>
<td>32</td>
<td>$\approx 6.4K$</td>
</tr>
<tr>
<td>+ DCNv2$^6$</td>
<td>33.2</td>
<td>35.8</td>
<td>31</td>
<td>$\approx 6.4K$</td>
</tr>
<tr>
<td>+ more anchors</td>
<td>33.5</td>
<td>36.3</td>
<td>30</td>
<td>$\approx 12.8K$</td>
</tr>
<tr>
<td>+ cosine annealing$^8$</td>
<td>34.8</td>
<td>37.9</td>
<td>30</td>
<td>$\approx 12.8K$</td>
</tr>
</tbody>
</table>

$^1$For all models, we follow and report the results on the mmdetection framework$^1$ on a single Tesla V100 GPU. Mmdetection’s YOLACT is slower than the official implementation by Bolya et al.$^2$, who reported 45fps.

$^2$Note that more efficient models can be obtained by using better anchor design methods$^3$.

$^3$We use 25fps as the cut-off for “real-time” following the common video signal standards (e.g. PAL$^4$ and SECAM$^5$) and existing methods$^5$.
Table 5: Comparison with SOTA on COCO test-dev. Our maYOLACT-700 establishes a new SOTA for real-time instance segmentation. * implies our implementation for YOLACT with ATSS w.IoU. When a paper does not report a performance measure, N/A is assigned and we reproduce the performance using its repository for completeness (shown by †).

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{S}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fps &lt; 25</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YOLACT-700 [8]</td>
<td>ResNet-101</td>
<td>31.2</td>
<td>50.6</td>
<td>32.8</td>
<td>12.1</td>
<td>33.3</td>
<td>47.1</td>
<td>ICCV 19</td>
</tr>
<tr>
<td>PolarMask [3]</td>
<td>ResNet-101</td>
<td>32.1</td>
<td>53.7</td>
<td>33.1</td>
<td>14.7</td>
<td>33.8</td>
<td>45.3</td>
<td>CVPR 20</td>
</tr>
<tr>
<td>PolarMask++ [9]</td>
<td>ResNet-101</td>
<td>33.8</td>
<td>57.5</td>
<td>34.6</td>
<td>16.6</td>
<td>35.8</td>
<td>46.2</td>
<td>TPAMI 21</td>
</tr>
<tr>
<td>RetinaMask [1]</td>
<td>ResNet-101</td>
<td>34.7</td>
<td>55.4</td>
<td>36.9</td>
<td>14.3</td>
<td>36.7</td>
<td>50.5</td>
<td>Preprint</td>
</tr>
<tr>
<td>Mask R-CNN [10]</td>
<td>ResNet-50</td>
<td>36.8</td>
<td>59.2</td>
<td>39.3</td>
<td>17.1</td>
<td>38.7</td>
<td>52.1</td>
<td>ICCV 17</td>
</tr>
<tr>
<td><strong>fps ≥ 25</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YOLACT-550 [8]</td>
<td>ResNet-50</td>
<td>28.2</td>
<td>46.6</td>
<td>29.2</td>
<td>9.2</td>
<td>29.3</td>
<td>44.8</td>
<td>ICCV 19</td>
</tr>
<tr>
<td>YOLACT-550*</td>
<td>ResNet-50</td>
<td>29.7</td>
<td>49.9</td>
<td>30.7</td>
<td>11.9</td>
<td>32.4</td>
<td>42.7</td>
<td>Baseline</td>
</tr>
<tr>
<td>Solov2-448 [12]</td>
<td>ResNet-50</td>
<td>34.0</td>
<td>54.0</td>
<td>36.1</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>NeurIPS 20</td>
</tr>
<tr>
<td>Solov2-448†</td>
<td>ResNet-50</td>
<td>34.0</td>
<td>54.0</td>
<td>36.0</td>
<td>10.3</td>
<td>36.3</td>
<td>53.8</td>
<td>NeurIPS 20</td>
</tr>
<tr>
<td>YOLACT-550++ [9]</td>
<td>ResNet-50</td>
<td>34.1</td>
<td>53.3</td>
<td>36.2</td>
<td>11.7</td>
<td>36.1</td>
<td>53.6</td>
<td>TPAMI 20</td>
</tr>
<tr>
<td>YOLACT-550++ [9]</td>
<td>ResNet-101</td>
<td>34.6</td>
<td>53.8</td>
<td>36.9</td>
<td>11.9</td>
<td>36.8</td>
<td>55.1</td>
<td>TPAMI 20</td>
</tr>
<tr>
<td>CenterMask-Lite† [13]</td>
<td>VoVNetV2-39</td>
<td>35.7</td>
<td>56.7</td>
<td>37.9</td>
<td><strong>18.4</strong></td>
<td>37.8</td>
<td>47.3</td>
<td>CVPR 20</td>
</tr>
<tr>
<td>CenterMask-Lite [13]</td>
<td>VoVNetV2-39</td>
<td>36.3</td>
<td>N/A</td>
<td>N/A</td>
<td>15.6</td>
<td>38.1</td>
<td>53.1</td>
<td>CVPR 20</td>
</tr>
<tr>
<td>Solov2-512†</td>
<td>ResNet-50</td>
<td>36.9</td>
<td>57.5</td>
<td>39.4</td>
<td>12.8</td>
<td>39.7</td>
<td><strong>57.1</strong></td>
<td>NeurIPS 20</td>
</tr>
<tr>
<td>Solov2-512‡</td>
<td>ResNet-50</td>
<td>37.1</td>
<td>57.7</td>
<td>39.7</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>NeurIPS 20</td>
</tr>
<tr>
<td>maYOLACT-550 (Ours)</td>
<td>ResNet-50</td>
<td>35.2</td>
<td>56.2</td>
<td>37.1</td>
<td>14.7</td>
<td>38.0</td>
<td>51.4</td>
<td></td>
</tr>
<tr>
<td>maYOLACT-700 (Ours)</td>
<td>ResNet-50</td>
<td>37.7</td>
<td><strong>59.4</strong></td>
<td><strong>39.9</strong></td>
<td>18.1</td>
<td><strong>40.8</strong></td>
<td>52.5</td>
<td></td>
</tr>
</tbody>
</table>

aches 59.4 wrt. common AP$_{50}$ metric with a gap of 1.7 AP$_{50}$ points compared to its closest real time counterpart (i.e. SOLOv2-512).

**Comparison with other methods.** Our maYOLACT is also competitive against slower methods (Table 5): It outperforms PolarMask++ [9], RetinaMask [11], Mask R-CNN [10] and TensorMask [1] while being faster. To illustrate, on Tesla V100 GPU, our maYOLACT-700 (i) has $\sim 2 \times$ more throughput with 25fps and nearly 4 mask AP gain (37.7 AP - Table 5) compared to PolarMask++ on ResNet-101 with 14 fps test time; and (ii) is $\sim 8 \times$ faster than TensorMask on ResNet-101 (i.e. $\sim 3$ fps) with similar performance.

## 5 Conclusion

We presented maIoU to assign a proximity value for an anchor compared to both a ground truth box and its mask. Using maIoU to assign anchors as positive or negative for training instance segmentation methods, we utilised the shape of objects as provided by the ground-truth segmentation masks. We showed that ATSS with our maIoU also improves throughput of the model. Exploiting this efficiency, we improved the performance further and reached SOTA results in real-time.

**Acknowledgements:** This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) (under grants 117E054 and 120E494). We also gratefully acknowledge the computational resources kindly provided by TÜBİTAK ULAKBIM High Performance and Grid Computing Center (TRUBA) and Roketsan Missiles Inc. Dr. Kalkan is supported by the BAGEP Award of the Science Academy, Turkey.
References


