# DANCE : A Deep Attentive Contour Model for Efficient Instance Segmentation

Zichen Liu[1,2*†] Jun Hao Liew[1*] Xiangyu Chen[2] Jiashi Feng[1]
[1] National University of Singapore [2] Shopee Data Science
`{liuzichen, liewjunhao}@u.nus.edu` `xiangyuwill@gmail.com` `elefjia@nus.edu.sg`

## Abstract

*Contour-based instance segmentation methods are attractive due to their efficiency. However, existing contour-based methods either suffer from lossy representation, complex pipeline or difficulty in model training, resulting in subpar mask accuracy on challenging datasets like MS-COCO. In this work, we propose a novel deep attentive contour model, named DANCE, to achieve better instance segmentation accuracy while remaining good efficiency. To this end, DANCE applies two new designs: **attentive contour deformation** to refine the quality of segmentation contours and **segment-wise matching** to ease the model training. Comprehensive experiments demonstrate DANCE excels at deforming the initial contour in a more natural and efficient way towards the real object boundaries. Effectiveness of DANCE is also validated on the COCO dataset, which achieves 38.1% mAP and outperforms all other contour-based instance segmentation models. To the best of our knowledge, DANCE is the first contour-based model that achieves comparable performance to pixel-wise segmentation models. Code is available at `https://github.com/lkevinzc/dance`.*

## 1. Introduction

Instance segmentation has received much research attention in the past few years due to its wide applications such as autonomous driving [20, 8, 30], robotic manipulation [22], etc. This task aims to locate all the objects in an image, classify them and produce segmentation masks delineating their shapes simultaneously. Instance segmentation is mostly formulated as a per-pixel (binary) classification problem within each region of interest (RoI) detected [10, 3, 15], which achieves good accuracy but often suffers heavy computation burden. Recently, some research works [23, 17, 31, 28, 29] consider instance segmentation as a contour vertices regression problem where each

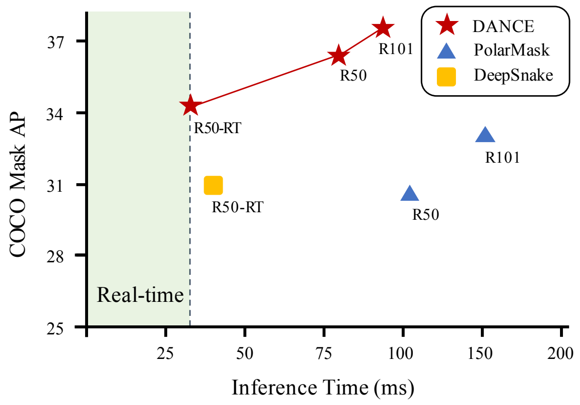


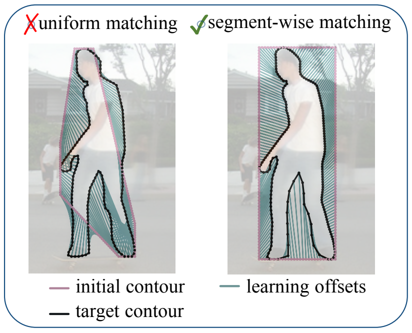
Figure 1: **Speed *vs*. Accuracy on COCO** `test-dev`. Inference time is measured using a single Tesla V100 GPU on the same machine. Our DANCE significantly outperforms all state-of-the-art contour-based instance segmentation methods [28, 23].

instance is represented by a closed contour. Compared with pixel-based approaches which require processing every single pixel within each detected RoI (*e.g.* $28 \times 28 = 784$ in Mask R-CNN [10]), contour-based methods enjoy better parametrization efficiency as they typically require much less parameters to represent the same mask (e.g. $128 \times 2 = 256$ values to represent the x, y coordinates of sampled contour vertices in DeepSnake [23]). Furthermore, contour-based models directly output the locations of predicted contour vertices and need not perform costly post-processing steps such as mask upsampling.

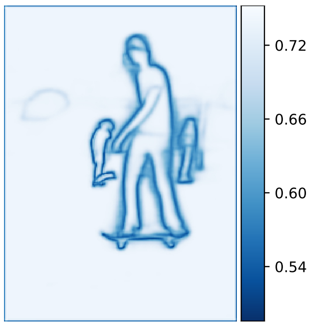
Although contour-based instance segmentation methods offer attractive efficiency, they usually struggle to yield accurate masks especially on challenging datasets like COCO [18] where large shape variations exist. For example, in [28, 29], mask contours are constructed from a set of end points of concentric rays emitted from object center, thus limiting the models to only handling convex shapes. Such lossy contour representation often suffers inevitable reconstruction error.

Some other contour-based approaches [31, 17, 23] accomplish segmentation by gradually deforming an initial

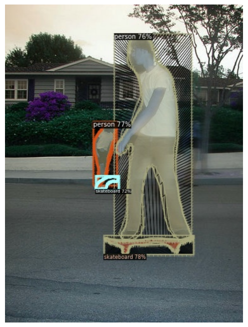
*Authors contributed equally.
†This work was mainly done during an internship at Shopee Data Science.



(a) Different Matching Schemes



(b) Edge Attention Map



(c) Sample Result

Figure 2: **(a)** Comparison between uniform matching [23] and our segment-wise matching. Uniform matching suffers from *correspondence interlacing*, which is caused by accumulated error of perimeter length difference between initial and target contours. **(b)** The learned attention map on which modulating coefficients are sampled to adjust contour deformation. **(c)**: Example showing learned contour deformation with our method.

contour towards the target object boundaries. This manner greatly alleviates the reconstruction error. Among them, the more recently published DeepSnake [23] demonstrates state-of-the-art performance on several datasets [5, 24, 9]. However, we argue that its regression target (per-vertex matching between uniformly sampled vertices on the initial and target contours) leads to *correspondence interlacing*, where the deformation paths from the initial to ground truth contour cross over one another. An example is shown in the left subfigure of Figure 2a, where the learning offsets (green lines) severely interlace near the person's leg, making the learning difficult. Moreover, DeepSnake is trained to regress all vertices on the contour, including those vertices that have already arrived at the object boundaries, degrading its learning effectiveness.

In this paper, we present a deep attentive contour model (abbreviated as DANCE) for instance segmentation, which well tackles the aforementioned challenges in DeepSnake. DANCE incorporates two novel components. First, it applies a novel **segment-wise matching** scheme, which adaptively splits the contour into multiple smaller segments where per-vertex matching is performed locally within each contour segment. This process is performed by first spotting the intersection points between the initial and target contours, and then using them as breaking points for mitigating the correspondence interlacing problem. As shown in the right subfigure of Figure 2a as well as Figure 2c, such a matching scheme enables a smoother and more natural deformation path from the initial contour to the target contour, hence easing the learning process. In addition, we also find this scheme pairs well with the bounding box initialized contour, offering faster processing speed compared to DeepSnake which requires converting the box into an octagon to serve as the initial contour for deformation.

Second, instead of regressing all vertices including those which have already arrived at the target boundaries, we further propose to apply an attention mechanism to the contour deformation such that the model can better focus on off-boundary vertices during deformation. We term this **attentive contour deformation**. In particular, we incorporate a lightweight attention module into DANCE, which learns a modulation coefficient for each vertex with guidance from the pixel-wise boundary prediction. An example of learned attention is shown in Figure 2b, where the instance contours are captured successfully. Therefore, as shown in Figure 2c, only vertices that are far away from the object boundaries are to be deformed and those on-boundary vertices are left untouched.

In summary, we make the following contributions:

- We propose a novel segment-based matching scheme that allows smoother deformation path for training;
- We design an attentive deformation mechanism to encourage the model to place more emphasis on off-boundary vertices during deformation;
- As shown in Figure 1, our DANCE demonstrates state-of-the-art performance among contour-based methods on COCO, achieving 38.1% Mask mAP.

## 2. Related Work

**Mask-based Instance Segmentation.** The majority of prior literature [3, 15, 10, 19] formulates instance segmentation as per-pixel classification inside RoIs. Mask R-CNN [10] is among the most representative works, which adds a parallel head to the existing detector [26] for non-competing pixel classification in a downsampled square region. In contrast to such a top-down paradigm, some other methods [8, 20] make image-wise prediction to embed each pixel first and then cluster or partition them into individual instance masks. However, these methods often require

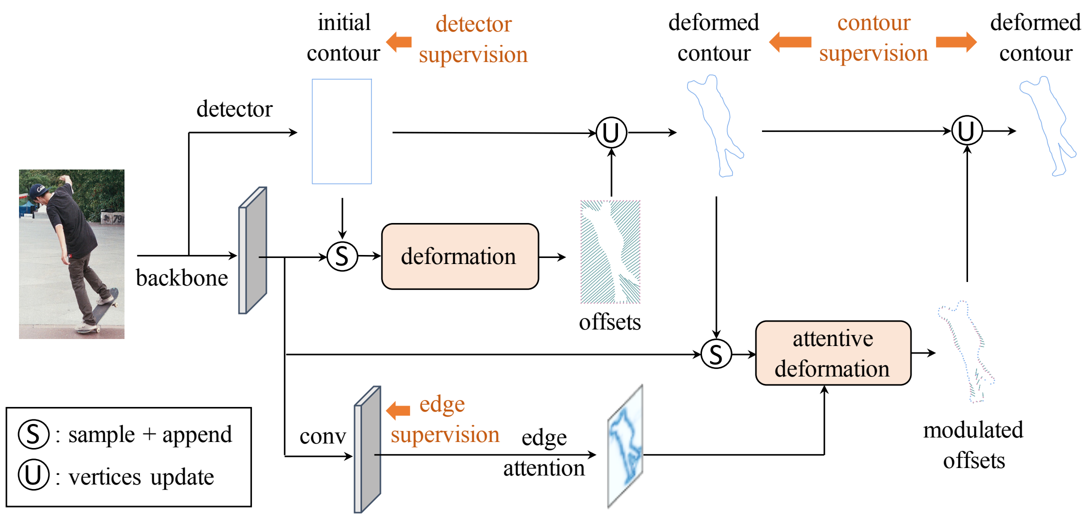


Figure 3: **Overall Pipeline of DANCE**. DANCE first samples on detected bounding boxes to obtain initial contours, which consist of a set of uniformly spaced points. The initial contour is first deformed to a coarse shape, and then the proposed attentive deformation is applied to those off-boundary points (purple) to refine and produce the final mask contour. For simplicity, we only show 2 `snake` modules in this figure.

costly postprocessing. In more recent literature, YOLACT [1] and BlendMask [2] propose to merge region-wise prediction and image-wise prediction through cropping or multiplication, both achieving better accuracy-speed trade-off.

**Contour-based Instance Segmentation.** Comparatively, the contour-based network is more lightweight. ESE-Seg [29] integrates an additional branch into YOLOv3 detector [25] to regress Chebyshev polynomial coefficients of inner-center radius. PolarMask [28] extends the anchor-free one-stage detector FCOS [27] to directly regress the lengths of rays at a uniform angle interval. They are both center-based and employ polar coordinate parameterization. Unfortunately, such methods usually suffer from reconstruction upper limit especially for non-convex shapes. There are also some point-based methods [17, 31, 23] that regard the contour as a polygon taking a set of points in the Cartesian coordinate system as vertices, and generate per-vertex offsets to refine the contour towards the real object boundaries. To anchor the initial contour vertices, Dense RepPoints [31] initializes at every location on the 2D feature maps; PolyTransform [17] employs an additional model to generate a coarse mask and samples vertices along the mask contour; DeepSnake [23] proposes to locate the extreme points first and generate an octagon on which contour vertices are uniformly sampled.

## 3. DANCE

We first revisit DeepSnake [23] based on which our method is developed. Then we explain our segment-wise matching scheme for enabling smooth contour deformation, followed by attentive deformation that adaptively modulates deformation offsets based on the guidance from pixel-wise boundary prediction. An overview of our DANCE is shown in Figure 3.

### 3.1. Preliminary: DeepSnake

DeepSnake [23] integrates the idea of classic snake algorithm [13] into the instance segmentation framework by making it learnable. The overall pipeline is as follows. Given a detected bounding box, the four center points on box edges are first located (also known as 'diamond' vertices according to [23]). Then, a deep `snake` module is employed to shift these four diamond vertices to extreme points [21] (*i.e.* top, bottom, leftmost and rightmost points), which are used to construct an octagon following [33]. $K$ vertices are uniformly sampled along the octagonal contour. Finally, three separate `snake` modules are used to process the $k$ vertices to iteratively refine the octagonal contour towards the real object boundaries.

To formalize, let $\{(x_k, y_k)\}_{k=1,\dots,K}$ denote $K$ uniformly sampled vertices on the initial contour, where $(x_k, y_k)$ refers to the the Cartesian coordinate of the $k$-th vertex. This is an ordered point set with the first point being the top extreme point. Then the `snake` module outputs per-vertex offsets as follows:

$$\{(\Delta x_k, \Delta y_k)\}_{k=1}^{K} = \texttt{snake}(\{\mathcal{F}(x_k, y_k)\}_{k=1}^{K}), \quad (1)$$

where the input feature $\mathcal{F}(x_k, y_k)$ is the concatenation of a learnable feature and the relative vertex coordinates $(x_k^{'}, y_k^{'})$. The appended relative coordinates are obtained by

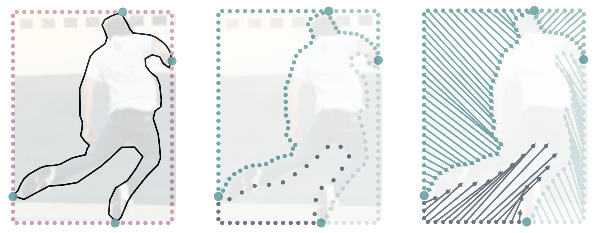

(1) uniformly sample on the box contour and locate intersections
(2) uniformly sample on the target segments broken by intersections
(3) perform segment-wise matching between initial and target contours

Figure 4: **Segment-wise Matching Scheme**. Purple dots in the left subfigure denote the sampled points along the initial contour. After splitting the segments by contour intersections, we denote the vertices in different segments with different colors in the middle and right subfigures, and then match them locally. In this illustrative example, the number of vertices $K$ is set to be 98 for better visualization purpose.

subtracting the vertex coordinates $(x_k, y_k)$ by the minimum value over all $k$ vertices to ensure the deformation is invariant to the translation of the contour. Given the predicted offsets by the `snake` module, each vertex is then updated as

$$\begin{bmatrix} x_k^{new} \\ y_k^{new} \end{bmatrix} = \begin{bmatrix} x_k^{old} \\ y_k^{old} \end{bmatrix} + \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix}. \qquad (2)$$

The `snake` module follows the structure of DeepGCNs [16] except that the GCN component is replaced with the circular convolution for aggregating information along the contour. In the original implementation, DeepSnake employs four `snake` modules, one for localization of extreme points given the four center points on the bounding box, and three others for iterative contour deformation from octagons towards object boundaries. We refer the interested readers to the original paper [23] for more details.

During training, for the ground truth contours, $K$ vertices are similarly uniformly sampled along the object boundaries with the first vertex being the one closest to the top extreme point in the initial octagon contour. The offset of each vertex is then assigned accordingly, which defines the learning target. Nevertheless, as shown in Figure 2a, we notice that such a naïve matching scheme completely disregards the initial contour shape when assigning the ground truth vertex offsets, which often results in correspondence interlacing, where the deformation paths from the initial to ground truth contours cross over one another. This leads to greater difficulty of learning in the `snake` module. In the next subsection, we introduce a novel segment-wise matching scheme to overcome this limitation.

### 3.2. Segment-wise Matching Scheme

As above discussed, the default matching scheme used in DeepSnake disregards the initial contour shape, which leads to a sub-optimal regression target for learning. To tackle this, we propose a novel segment-wise matching scheme where the initial contour is first divided into multiple segments and the assignment of ground truth vertices is performed locally within each segment. In this way, the correspondence interlacing phenomenon can be effectively alleviated, yielding a much smoother deformation path for learning. However, the criteria for contour splitting still remains unclear. In this work, we propose a piece of simple heuristics for splitting the contour. In particular, we first look for the intersection points of ground truth contours and initial contours up to some tolerance margin [1] Then the ground truth contour can be decomposed into several contour segments. Within each segment, we re-sample the ground truth contour uniformly based on the number of vertices on the corresponding segment of the initial contour. Figure 4 depicts the process of such a segment-wise matching scheme. With segment-wise matching, the initial bounding box already suffices and it becomes unnecessary to use an octagon as the initial contour. The additional `snake` module used in DeepSnake for locating extreme points to construct octagons can thus be eliminated, leading to faster training and inference speed.

### 3.3. Attentive Deformation

Given ground truth vertex-wise offsets, DeepSnake is trained to regress all vertices on the contour, including those close to the object boundaries. However, such formulation requires the model to learn to simultaneously regress both the on- and off-boundary vertices, which have very different dynamics that would make the learning less effective. Instead, we propose to decompose the regression process into two components: 1) the offset regression similar as before, and 2) an attention module that looks for the off-boundary points among all the vertices, forcing the `snake` module to focus on regressing those vertices during deformation while leaving the others untouched. Altogether, we term this the attentive deformation mechanism. To realize it, we propose a simple yet effective attention module (Figure 5) which outputs an attention map $F_{att}$ that adaptively re-weights the offsets predicted by the `snake` module as follows:

$$\begin{bmatrix} \Delta x_k^{'} \\ \Delta y_k^{'} \end{bmatrix} = F_{att}(x_k, y_k) * \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix}. \qquad (3)$$

To further enhance the capability of the attention module when assessing whether a vertex already reaches the object boundary, we explicitly incorporate object boundaries information into the attention module. Specifically, given a feature map extracted from the CNN, we inject class-agnostic boundary prediction supervision on the feature map using

[1] In the special case where bounding boxes are taken as the initial contours, these intersection points are extreme points

a $1 \times 1$ convolution with sigmoid activation. The resulting edge map is then passed to a few more convolutional layers interleaved with GroupNorm and ReLU, followed by a sigmoid activation function to produce an image level attention map, based on which per-pixel modulation coefficients are sampled and multiplied with the originally predicted offsets as in Equation (3).

## 3.4. Better Contour Normalization

The original implementation of DeepSnake involves normalizing the contour by subtracting each vertex coordinate by the minimum value over all vertices on the contour to enable translation-invariant contour deformation. However, we argue that such a translation-invariant normalization is not sufficient for stable learning, as the range of regression offsets heavily depends on the object size, which inevitably complicates the learning of the snake module. We address this issue by introducing a scale- and translation-invariant normalization scheme. Specifically, each vertex coordinate is normalized in the following manner before appended as the input feature of the snake module:

$$x_k^{'} = \frac{x_k - x_{\min}}{x_{\max} - x_{\min}}, \qquad y_k^{'} = \frac{y_k - y_{\min}}{y_{\max} - y_{\min}}, \tag{4}$$

where $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$ denote the minimum and maximum values over all vertices for a specific object contour, respectively. Since the input vertex coordinates are now bounded to be within $[0, 1]$, the resulting offset predictions should also be bounded. Therefore, we add a tanh activation function to the snake prediction before multiplying it by the object scale. Thus, Equations (1) and (2) are adapted as follows:

$$\begin{aligned} &\{(\Delta x_k, \Delta y_k)\}_{k=1}^{K} = \tanh\left(\texttt{snake}(\{\mathcal{F}(x_k, y_k)\}_{k=1}^{K})\right); \\ &x_k^{new} = x_k^{old} + w * \Delta x_k, \quad y_k^{new} = y_k^{old} + h * \Delta y_k, \end{aligned} \tag{5}$$

where $w$ and $h$ denote the width and height of the object. Interestingly, the proposed normalization scheme plays a similar role as the RoIAlign [10] operation used in mask-based approaches which projects objects with varying size into a fixed-sized feature map. In this way the model can focus on learning to accurately delineate the object shape regardless of its size and location in the image.

## 4. Experiments

Comprehensive experiments were conducted on two commonly used instance segmentation benchmarks.

- **MS COCO 2017 [18]** consists of 118*k* training, 5*k* validation and 20*k* testing images with 80 object categories. We use the COCO average precision (AP) metric to evaluate our method.

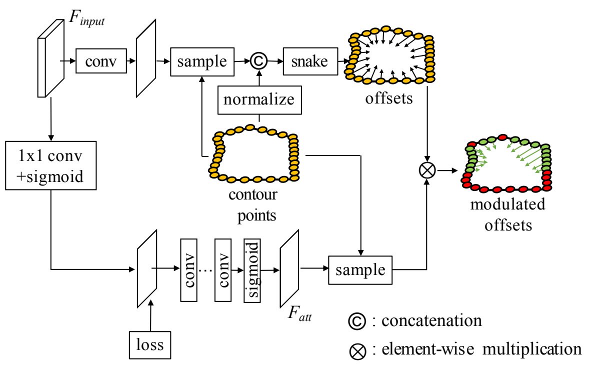


Figure 5: **Attentive Deformation Mechanism**. The $F_{input}$ is from FPN features; the current contour points are used to extract point features as well as modulating coefficients, which will be multiplied to get the final offsets result.

- **SBD [9]** contains 11,355 images with instance-level boundary annotations of 20 classes. It is split into 5,623 training images and 5,732 validation images. Following [23], we kept 500 images from the validation set as our mini-val set and only tested on the complete validation set once for the final performance report. Results are evaluated in terms of 2010 VOC [9] AP$_{vol}$, AP$_{50}$ and AP$_{70}$.

## 4.1. Implementation Details

**Detector.** While our method does not depend on any specific detector, we adopt both CenterNet [32] and FCOS [27] for initializing mask contours. The former one allows fair comparison with DeepSnake[2] [23], while the latter one is more powerful to handle challenging dataset like COCO. In addition, we also re-implemented DeepSnake with FCOS as the object detector for fair comparison.

**Attentive Deformation.** Following DeepSnake [23], we also perform three iterations of contour deformation by applying three individual snake modules. We only apply attentive deformation to the last two snake modules since most vertices on the initial contour are far from the true object boundaries. We take the P2 feature from FPN as input for attentive deformation ($F_{input}$ in Figure 5).

**Losses.** In addition to the detection losses used in the object detector, the vertex-wise offset prediction is trained using a smooth $l$-1 loss. The loss is averaged across $K$ sampled points. Unless specified otherwise, we set $K$ to 196 in all experiments, which can cover most object shapes. We notice that the loss is sensitive to the object size as the object with larger size tends to produce larger offsets. To stabilize the learning, we weight the loss of the individual instance

[2]DeepSanke bases their experiments on CenterNet

| ID | Initial | Matching | Attentive | Normalized | MS-Feature | AP | AP$_{50}$ | AP$_{75}$ | AP$_s$ | AP$_m$ | AP$_l$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | octagon | uniform | | | | 30.6 | 51.4 | 31.6 | 15.1 | 33.3 | 43.0 |
| M1 | octagon | uniform | | | | 32.1 | 53.5 | 33.4 | 16.1 | 34.9 | 45.1 |
| M2 | octagon | segment | | | | 32.9 | 54.3 | 34.4 | 16.3 | 35.6 | 46.5 |
| M3 | box | segment | | | | 32.7 | 54.0 | 33.7 | 16.2 | 36.9 | 45.5 |
| M4 | box | segment | ✓ | | | 33.4 | 54.9 | 34.8 | 16.7 | 36.6 | 46.3 |
| M5 | box | segment | | ✓ | | 33.4 | 54.3 | 35.1 | 16.5 | 36.4 | 47.1 |
| M6 | box | segment | ✓ | ✓ | | 34.2 | 55.1 | 35.9 | 16.9 | 37.4 | 47.6 |
| M7 | box | segment | ✓ | ✓ | ✓ | **34.5** | **55.3** | **36.5** | **17.2** | **37.5** | **48.0** |

Table 1: **Ablation Results**. The first two rows denote our baseline models. Model M0 is a direct combination of FCOS and DeepSnake; model M1 is our augmented baseline which employs a modified FCOS (with direct extreme points prediction in an anchor-free manner) to efficiently generate initial octagon contours; model M7 is our finalized model. Performance is evaluated on MS COCO `val2017`.

| Matching | AP | AP$_{50}$ | AP$_{75}$ | AP$_s$ | AP$_m$ | AP$_l$ |
|---|---|---|---|---|---|---|
| Chamfer | 27.4 | 49.8 | 26.5 | 13.7 | 30.3 | 38.7 |
| uniform | 31.4 | 52.9 | 32.5 | 15.9 | 34.5 | 43.6 |
| **segment** | **32.7** | **54.0** | **33.7** | **16.2** | **36.9** | **45.5** |

Table 2: **Different Matching Methods**. Both unordered (Chamfer loss) and ordered (either uniformly sampled or segment-wise sampled) matching methods are evaluated using object boxes as initial contours. Chamfer loss seeks the nearest points to apply distance penalty, which could not ensure point correspondences.

by the inverse of its bounding box's side lengths. Mathematically, the loss for training each `snake` module is

$$\mathcal{L}_{snake} = \frac{1}{K}\sum_{k=1}^{K} \text{smooth}_{L_1}\left(\begin{bmatrix}\tilde{x_k}/w \\ \tilde{y_k}/h\end{bmatrix} - \begin{bmatrix}x_k^*/w \\ y_k^*/h\end{bmatrix}\right), \quad (6)$$

where $\tilde{x_k}, \tilde{y_k}$ denote the updated contour vertices using Equation (5) whereas $x_k^*, y_k^*$ are the target vertex locations. Following [7], we employ Dice loss for supervising the boundary prediction in attentive deformation. The overall loss is as follows:

$$\mathcal{L} = \mathcal{L}_{det} + \alpha\mathcal{L}_{snake} + \beta\mathcal{L}_{edge}, \quad (7)$$

where $\alpha$ and $\beta$ are empirically set to 10 and 1, respectively.

**Training & Testing.** For COCO dataset, we employ ResNet-50 [11] pre-trained on ImageNet [6] as the backbone and train for 12 epochs (1× schedule) for all experiments unless otherwise specified. All models are trained using stochastic gradient descent with learning rate of 0.005. The weight decay and momentum are set to be 0.0001 and 0.9, respectively. The learning rate is decreased by 10× at iteration 120k and 160k. Random horizontal flipping and multi-scale training is adopted while no testing augmentation is used. For the SBD dataset, we exactly follow the training and testing settings of DeepSnake [23] so that all other factors remain unchanged.

## 4.2. Ablation Results

In order to justify our design choices, we conduct ablation study on COCO dataset to quantitatively analyze the effect of each component. Results are given in Table 1.

**Baselines.** The model M0 denotes the vanilla baseline model, which directly combines FCOS [27] with DeepSnake [23]. Model M1 is our augmented baseline which employs a modified FCOS to efficiently generate initial octagon contours. We extend the bounding box regression branch in FCOS to predict four additional relative gliding offsets in an anchor-free manner. It is supervised via smooth $l$-1 loss, which is appended to the total loss with weight 1. We empirically find this simple modification is more effective than introducing an additional `snake` module as used in DeepSnake [23] and saves about 7*ms* computation time.

**Matching.** We also compare our proposed segment-based matching scheme with 'Chamfer' (applying Chamfer loss between two unordered point sets) and uniform matching (smooth $l$-1 loss on orderly matched points). As shown in Table 2, our segment-based matching scheme yields the best performance. The ablation in Table 1 also shows replacing uniform matching scheme used in [23] by our proposed segment-wise matching (M1 → M2) offers 0.8% AP improvement. Moreover, the learned deformation path is more natural thus yielding better mask quality as compared in Figure 6. When comparing models M2 and M3, we can see that our segment-wise matching scheme is robust to different contour initialization (octagon and box contour). In the remaining experiments, we use bounding box as our initial contour due to its faster speed while sampling.

**Attentive Deformation.** Comparing model M4 and M3, our proposed attentive deformation mechanism further improves the performance by 0.7% AP, demonstrating the importance of decoupling the regression task into offset learning and classification of on- and off-boundary vertices.

**Contour Normalization.** We can see that a scale- and

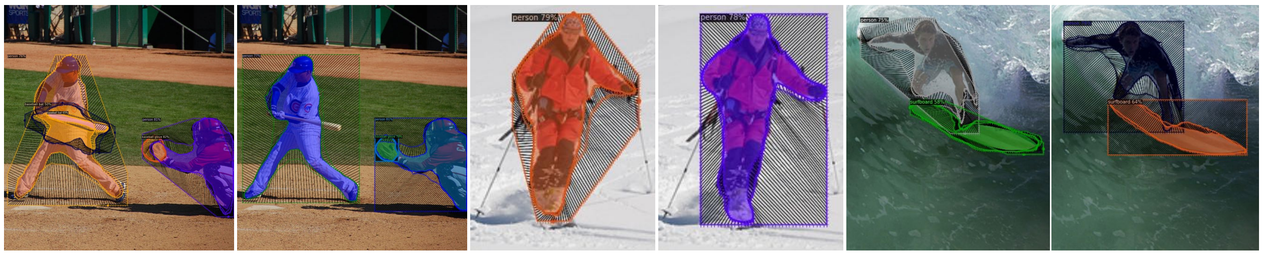


Figure 6: **Example Result Pairs**. We compare deformation paths learned from uniform matching [23] (left) and our segment-wise matching (right). The left images are from the baseline model M1 and right ones are from model M3 as numbered in Table 1. The lines show the initial and intermediate contour results, while the final mask results are overlaid. Our method produces more natural deformation paths and hence higher quality masks. Better viewed in color pdf with zoom-in.

| Method | Backbone | Epoch | Aug. | AP | $AP_{50}$ | $AP_{75}$ | $AP_s$ | $AP_m$ | $AP_l$ |
|---|---|---|---|---|---|---|---|---|---|
| *pixel-based* | | | | | | | | | |
| Mask R-CNN* [10] | R-50-FPN | 24 | ○ | 34.9 | 56.8 | 36.8 | 15.1 | 36.7 | 50.6 |
| Mask R-CNN* | R-101-FPN | 72 | ✓ | 38.3 | 61.2 | 40.8 | 18.2 | 40.6 | 54.1 |
| BlendMask [2] | R-50-FPN | 36 | ✓ | 37.0 | 58.9 | 39.7 | 17.3 | 39.4 | 52.5 |
| BlendMask | R-101-FPN | 36 | ✓ | 38.4 | 60.7 | 41.3 | 18.2 | 41.5 | 53.3 |
| CenterMask [15] | R-101-FPN | 24 | ✓ | 38.3 | - | - | 17.7 | 40.8 | 54.5 |
| *contour-based* | | | | | | | | | |
| ExtremeNet [33] | Hourglass-104 | 100 | ✓ | 18.9 | 44.5 | 13.7 | 10.4 | 20.4 | 28.3 |
| Dense RepPoints [31] | X-101-DCN | 12 | ✓ | 30.0 | 57.2 | 28.2 | - | - | - |
| PolarMask [28] | R-101-FPN | 12 | ○ | 30.4 | 51.9 | 31.0 | 13.4 | 32.4 | 42.8 |
| DeepSnake [23] | DLA-34 | 160 | ✓ | 30.3 | - | - | - | - | - |
| **DANCE** | R-50-FPN | 12 | ✓ | 34.6 | 55.9 | 36.4 | 19.3 | 37.2 | 43.9 |
| **DANCE** | R-50-FPN | 36 | ✓ | 36.8 | 58.5 | 39.0 | 21.1 | 39.1 | 46.9 |
| **DANCE** | R-101-FPN | 24 | ✓ | 38.1 | 60.2 | 40.5 | 21.5 | 40.7 | 48.8 |

Table 3: **Quantitative Results on MS COCO [18]**. We compare our DANCE with state-of-the-art models on `test-dev`. 'R' and 'X' denote ResNet and ResNeXt, respectively. 'Aug.' denotes using multi-scale augmentation for training. Mask R-CNN* refers to re-implementation in [4].

translation-invariant normalization scheme outperforms the default translation-invariant version as used in DeepSnake by 0.8% AP (model M6 *v.s.* model M4). Since objects have different sizes, the model needs to predict offsets with high diversity, which means difficulty in optimization. With object scales normalized, our `snake` modules can focus more on learning to accurately delineate the object shapes.

**Multi-scale Feature.** We also experiment by replacing the $F_{input}$ with a multi-scale feature for stronger representation. Specifically, we apply feature pyramid fusion following [14] to fuse the FPN features `P2` to `P5`. As shown in the last row of Table 1, this further improves the performance by 0.3% AP as compared to using `P2` alone as the input for the `snake` module, demonstrating the effectiveness of multi-scale features for prediction of deformation offsets.

### 4.3. Comparison with State-of-the-Arts

**MS-COCO.** We compare our DANCE with the state-of-the-art instance segmentation methods on COCO `test-dev`. As shown in Table 3, we first observe our DANCE with ResNet-50 as backbone trained for $1\times$ schedule (34.6% AP on `test-dev`) already surpasses all other contour-based methods by a large margin. Moreover, it also achieves competitive performance with pixel-based methods such as Mask R-CNN [10] or BlendMask [2]. Since the models in ablation study are under-fitted, we extend the training to $3\times$ schedule and apply stronger ResNet-101 backbone, boosting the performance to 38.1%AP, which is comparable with the top performing pixel-based methods. Some qualitative results are presented in Figure 7.

**SBD.** We also compare our DANCE with top-performing contour-based segmentation methods on the SBD dataset. We adopt the same backbone, detector, training and testing configuration as DeepSnake [23], and validate that our method outperforms all other methods (Table 4).

**Real-time Setting.** Moreover, we construct a real-time version of our DANCE model, called DANCE-RT, for com-

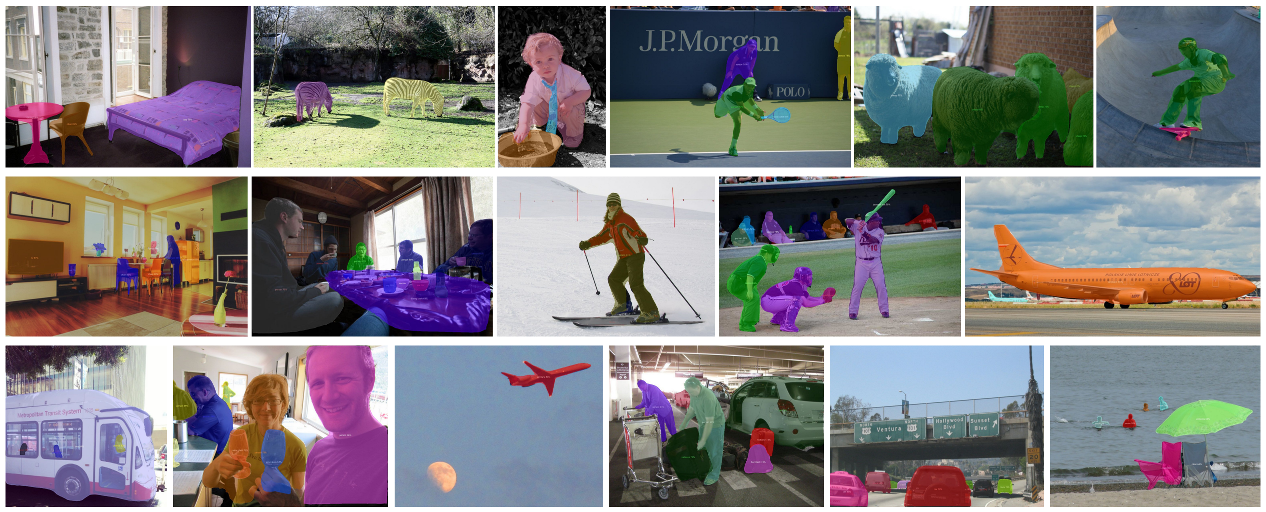


Figure 7: **Qualitative Results**. Our DANCE model produces high quality segmentation results on COCO `val` images.

| Method | AP$_{vol}$ | AP$_{50}$ | AP$_{70}$ | FPS |
|---|---|---|---|---|
| Embedding-20 [12] | 31.5 | 34.6 | 15.0 | 35.7 |
| ESE-Seg-20 [29] | 35.3 | 40.7 | 12.1 | 41.7 |
| DeepSnake [23] | 54.4 | 62.1 | 48.3 | 34.6 |
| DANCE | **56.2** | **63.6** | **50.4** | 38.3 |

Table 4: **Results on SBD Validation Set**. We compare our DANCE with top contour-based methods on SBD validation set. Results show that our methods outperform them by a large margin. Inference time for DeepSnake and DANCE is measured using a single V100 GPU on the same machine to give direct comparison, while that for the other methods is cited from the corresponding paper.

| Method | Backbone | Time (ms) | AP (test-dev) | AP (val) |
|---|---|---|---|---|
| ESE-Seg [29] | Dark-53 | 26.0 | - | 21.6 |
| YOLACT [1] | R-101 | 34.2 | 29.8 | 29.9 |
| DeepSnake [23] | DLA-34 | 36.8 | 30.3 | 30.5 |
| DeepSnake* | R-50 | 37.3 | 31.0 | 30.6 |
| **DANCE-RT**† | R-50 | 33.8 | 34.1 | 33.8 |
| **DANCE-RT** | R-50 | 35.2 | **34.6** | **34.2** |

Table 5: **Real-time Setting Results**. We compare our DANCE with other real-time instance segmentation models on COCO `val` and `test-dev`, proving its real-time effectiveness. † denotes our DANCE-RT model without the last `snake` module for faster inference speed. * denotes the DeepSnake model with exactly the same backbone and detector as our DANCE-RT.

parison with other real-time instance segmentation models. Inspired by FCOS-RT [27], we first make the detector lighter by using only 3 FPN levels (P3∼P5). Instead of using the same structure for all the `snake` modules as in DeepSnake [23], we customize each `snake` module with different atrous circular convolution rates and numbers of layers. Specifically, the earlier `snake` modules responsible for providing large deformation are assigned with larger atrous rates and bigger numbers of layers as compared to the latter ones which aim at fine-grained refinement. Due to limited space, we refer the readers to the supplementary material for more details. The number of sampling points is decreased to 96 to reduce computation. We adopt 4× training schedule and downsize the input images for both training and testing. During inference, we can optionally remove the last `snake` module to trade off between speed and accuracy. As reported in Table 5, DANCE-RT† gets 1.4 $ms$ speed-up at the cost of 0.5% AP drop. Table 5 also demonstrates that our DANCE-RT outperforms other real-time instance segmentation methods in terms of both accuracy and speed, except ESE-Seg [29] which has much inferior segmentation accuracy as compared to ours.

## 5. Conclusion

In this work, we presented two novel designs, attentive deformation mechanism and segment-wise matching scheme, that are essential for contour-based instance segmentation models to learn more effective deformation. Our best model achieves 38.1% AP on COCO `test-dev`, which is comparable to existing top-performing pixel-based models. We also proposed a real-time variant with good speed-accuracy trade-off, demonstrating its potential for real-time applications.

**Acknowledgement.** Jiashi Feng was partially supported by AISG-100E-2019-035, MOE2017-T2-2-151, NUS_ECRA_FY17_P08 and CRP20-2017-0006.

## References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *International Conference on Computer Vision*, 2019.

[2] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[3] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *International Conference on Computer Vision (ICCV)*, 2019.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[7] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *European Conference on Computer Vision*, 2018.

[8] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *International Conference on Computer Vision*, 2019.

[9] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision*, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[12] Saumya Jetley, Michael Sapienza, Stuart Golodetz, and Philip H. S. Torr. Straight to shapes: Real-time detection of encoded shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[13] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. In *International Journal of Computer Vision*, 1988.

[14] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[16] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *International Conference on Computer Vision (ICCV)*, 2019.

[17] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[19] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[20] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[21] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *International Conference on Computer Vision (ICCV)*, 2017.

[22] Deepak Pathak, Yide Shentu, Dian Chen, Pulkit Agrawal, Trevor Darrell, Sergey Levine, and Jitendra Malik. Learning instance segmentation by interaction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2042–2045, 2018.

[23] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[24] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3009–3018, 2019.

[25] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Conference on Neural Information Processing Systems*, 2015.

[27] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *International Conference on Computer Vision*, 2019.

[28] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[29] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[30] Xingqian Xu, Mang Tik Chiu, Thomas S Huang, and Honghui Shi. Deep affinity net: Instance segmentation via affinity. *arXiv preprint arXiv:2003.06849*, 2020.

[31] Ze Yang, Yinghao Xu, Han Xue, Zheng Zhang, Raquel Urtasun, Liwei Wang, Stephen Lin, and Han Hu. Dense reppoints: Representing visual objects with dense point sets. *https://arxiv.org/abs/1912.11473v1*, 2019.

[32] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[33] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.