SOLO: Segmenting Objects by Locations*

Xinlong Wang[1], Tao Kong[2], Chunhua Shen[1], Yuning Jiang[2], and Lei Li[2]

[1] The University of Adelaide, Australia
[2] ByteDance AI Lab

Abstract. We present a new, embarrassingly simple approach to instance segmentation. Compared to many other dense prediction tasks, *e.g.*, semantic segmentation, it is the arbitrary number of instances that have made instance segmentation much more challenging. In order to predict a mask for each instance, mainstream approaches either follow the "detect-then-segment" strategy (*e.g.*, Mask R-CNN), or predict embedding vectors first then use clustering techniques to group pixels into individual instances. We view the task of instance segmentation from a completely new perspective by introducing the notion of "instance categories", which assigns categories to each pixel within an instance according to the instance's location and size, thus nicely converting instance segmentation into a single-shot classification-solvable problem. We demonstrate a much simpler and flexible instance segmentation framework with strong performance, achieving *on par* accuracy with Mask R-CNN and outperforming recent single-shot instance segmenters in accuracy. We hope that this simple and strong framework can serve as a baseline for many instance-level recognition tasks besides instance segmentation. Code is available at `https://git.io/AdelaiDet`

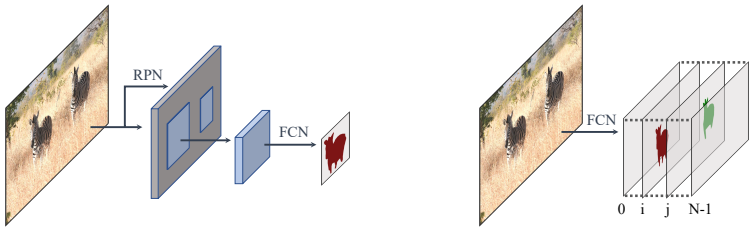
Keywords: Instance segmentation, Location category

1 Introduction

Instance segmentation is challenging because it requires the correct separation of all objects in an image while also semantically segmenting each instance at the pixel level. Objects in an image belong to a fixed set of semantic categories, but the number of instances varies. As a result, semantic segmentation can be easily formulated as a dense per-pixel classification problem, while it is challenging to predict instance labels directly following the same paradigm.

To overcome this obstacle, recent instance segmentation methods can be categorized into two groups, *i.e.*, top-down and bottom-up paradigms. The former approach, namely 'detect-then-segment', first detects bounding boxes and then segments the instance mask in each bounding box. The latter approach learns an affinity relation, assigning an embedding vector to each pixel, by pushing away pixels belonging to different instances and pulling close pixels in the same instance. A grouping post-processing is then needed to separate instances. Both

* Correspondence should be addressed to C. Shen.



(a) Mask R-CNN (b) SOLO

Fig. 1. Comparison of the pipelines of Mask R-CNN and the proposed SOLO.

these two paradigms are step-wise and *indirect*, which either heavily rely on accurate bounding box detection or depend on per-pixel embedding learning and the grouping processing.

In contrast, we aim to directly segment instance masks, under the supervision of full instance mask annotations instead of masks in boxes or additional pixel pairwise relations. We start by rethinking a question: *What are the fundamental differences between object instances in an image?* Take the challenging MS COCO dataset [16] for example. There are in total 36,780 objects in the validation subset, 98.3% of object pairs have center distance greater than 30 pixels. As for the rest 1.7% of object pairs, 40.5% of them have size ratio greater than 1.5×. To conclude, in most cases two instances in an image either have different center locations or have different object sizes. This observation makes one wonder whether we could directly distinguish instances by the center locations and object sizes?

In the closely related field, semantic segmentation, now the dominate paradigm leverages a fully convolutional network (FCN) to output dense predictions with N channels. Each output channel is responsible for one of the semantic categories (including background). Semantic segmentation aims to distinguish different semantic categories. Analogously, in this work, we propose to distinguish object instances in the image by introducing the notion of "*instance categories*", *i.e.*, the quantized center locations and object sizes, which enables to **s**egment **o**bjects by **lo**cations, thus the name of our method, **SOLO.**

Locations An image can be divided into a grid of $S \times S$ cells, thus leading to S^2 center location classes. According to the coordinates of the object center, an object instance is assigned to one of the grid cells, as its center location category. Note that grids are used conceptually to assign location category for each pixel. Each output channel is responsible for one of the center location categories, and the corresponding channel map should predict the instance mask of the object belonging to that location. Thus, structural geometric information is naturally preserved in the spatial matrix with dimensions of height by width. Unlike DeepMask [24] and TensorMask [4], which run in a dense sliding-window manner and segment an object in a fixed local patch, our method naturally outputs accurate

masks for all scales of instances without the limitation of (anchor) box locations and scales.

In essence, an instance location category approximates the location of the object center of an instance. Thus, by classification of each pixel into its instance location category, it is equivalent to predict the object center of each pixel in the latent space. The importance here of converting the location prediction task into classification is that, with classification it is much more straightforward and easier to model varying number of instances using a fixed number of channels, at the same time not relying on post-processing like grouping or learning embeddings.

Sizes To distinguish instances with different object sizes, we employ the feature pyramid network (FPN) [14], so as to assign objects of different sizes to different levels of feature maps. Thus, all the object instances are separated regularly, enabling to classify objects by "instance categories". Note that FPN was designed for the purposes of detecting objects of different sizes in an image.

In the sequel, we empirically show that FPN is one of the core components for our method and has a profound impact on the segmentation performance, especially objects of varying sizes being presented.

With the proposed SOLO framework, we are able to optimize the network in an end-to-end fashion for the instance segmentation task using mask annotations solely, and perform pixel-level instance segmentation out of the restrictions of local box detection and pixel grouping. For the first time, we demonstrate a very simple instance segmentation approach achieving *on par* results to the dominant "detect-then-segment" method on the challenging COCO dataset [16] with diverse scenes and semantic classes. Additionally, we showcase the generality of our framework via the task of instance contour detection, by viewing the instance edge contours as a one-hot binary mask, with almost no modification SOLO can generate reasonable instance contours. The proposed SOLO only needs to solve two pixel-level classification tasks, thus it may be possible to borrow some of the recent advances in semantic segmentation for improving SOLO. *The embarrassing simplicity and strong performance of the proposed SOLO method may predict its application to a wide range of instance-level recognition tasks.*

1.1 Related Work

We review some instance segmentation works that are closest to ours.

Top-down Instance Segmentation. The methods that segment object instance in a priori bounding box fall into the typical top-down paradigm. FCIS [13] assembles the position-sensitive score maps within the region-of-interests (ROIs) generated by a region proposal network (RPN) to predict instance masks. Mask R-CNN [9] extends the Faster R-CNN detector [25] by adding a branch for segmenting the object instances within the detected bounding boxes. Based on Mask R-CNN, PANet [19] further enhances the feature representation to improve the accuracy, Mask Scoring R-CNN [10] adds a mask-IoU branch to predict the quality of the predicted mask and scoring the masks to improve the performance. HTC [2] interweaves box and mask branches for a joint multi-stage processing.

TensorMask [4] adopts the dense sliding window paradigm to segment the instance in the local window for each pixel with a predefined number of windows and scales. In contrast to the top-down methods above, our SOLO is totally box-free thus not being restricted by (anchor) box locations and scales, and naturally benefits from the inherent advantages of FCNs.

Bottom-up Instance Segmentation. This category of the approaches generate instance masks by grouping the pixels into an arbitrary number of object instances presented in an image. In [22], pixels are grouped into instances using the learned associative embedding. A discriminative loss function [7] learns pixel-level instance embedding efficiently, by pushing away pixels belonging to different instances and pulling close pixels in the same instance. SGN [18] decomposes the instance segmentation problem into a sequence of sub-grouping problems. SSAP [8] learns a pixel-pair affinity pyramid, the probability that two pixels belong to the same instance, and sequentially generates instances by a cascaded graph partition. Typically bottom-up methods lag behind in accuracy compared to top-down methods, especially on the dataset with diverse scenes. Instead of exploiting pixel pairwise relations SOLO directly learns with the instance mask annotations solely during training, and predicts instance masks end-to-end without grouping post-processing.

Direct Instance Segmentation. To our knowledge, no prior methods directly train with mask annotations solely, and predict instance masks and semantic categories in one shot without the need of grouping post-processing. Several recently proposed methods may be viewed as the 'semi-direct' paradigm. AdaptIS [26] first predicts point proposals, and then sequentially generates the mask for the object located at the detected point proposal. PolarMask [28] proposes to use the polar representation to encode masks and transforms per-pixel mask prediction to distance regression. They both do not need bounding boxes for training but are either being step-wise or founded on compromise, *e.g.*, coarse parametric representation of masks. Our SOLO takes an image as input, directly outputs instance masks and corresponding class probabilities, in a fully convolutional, box-free and grouping-free paradigm.

2 Our Method: SOLO

2.1 Problem Formulation

The central idea of SOLO framework is to reformulate the instance segmentation as two simultaneous category-aware prediction problems. Concretely, our system divides the input image into a uniform grids, *i.e.*, $S \times S$. If the center of an object falls into a grid cell, that grid cell is responsible for 1) predicting the semantic category as well as 2) segmenting that object instance.

Semantic Category For each grid, our SOLO predicts the C-dimensional output to indicate the semantic class probabilities, where C is the number of classes. These probabilities are conditioned on the grid cell. If we divide the input image

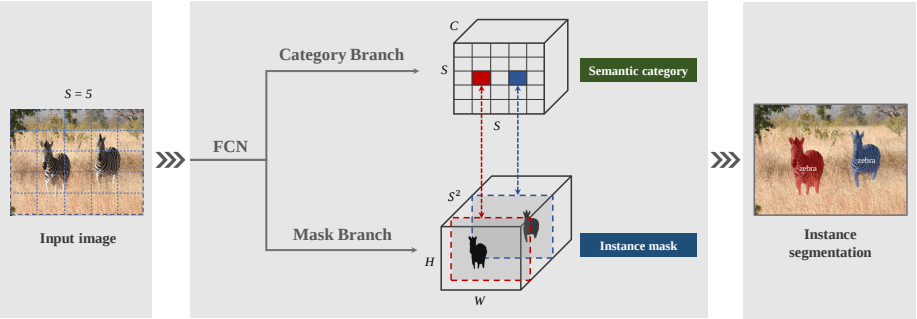


Fig. 2. SOLO framework. We reformulate the instance segmentation as two sub-tasks: category prediction and instance mask generation problems. An input image is divided into a uniform grids, *i.e.*, $S \times S$. Here we illustrate the grid with $S = 5$. If the center of an object falls into a grid cell, that grid cell is responsible for predicting the semantic category (top) and masks of instances (bottom). We do not show the feature pyramid network (FPN) here for simpler illustration.

into $S \times S$ grids, the output space will be $S \times S \times C$, as shown in Figure 2 (top). This design is based on the assumption that each cell of the $S \times S$ grid must belong to one individual instance, thus only belonging to one semantic category. During inference, the C-dimensional output indicates the class probability for each object instance.

Instance Mask In parallel with the semantic category prediction, each positive grid cell will also generate the corresponding instance mask. For an input image I, if we divide it into $S \times S$ grids, there will be at most S^2 predicted masks in total. We explicitly encode these masks at the third dimension (channel) of a 3D output tensor. Specifically, the instance mask output will have $H_I \times W_I \times S^2$ dimension. The k^{th} channel will be responsible to segment instance at grid (i, j), where $k = i \cdot S + j$ (with i and j zero-based)[3]. To this end, a one-to-one correspondence is established between the semantic category and class-agnostic mask (Figure 2).

A direct approach to predict the instance mask is to adopt the fully convolutional networks, like FCNs in semantic segmentation [20]. However the conventional convolutional operations are *spatially invariant* to some degree. Spatial invariance is desirable for some tasks such as image classification as it introduces robustness. However, here we need a model that is *spatially variant*, or in more precise words, position sensitive, since our segmentation masks are conditioned on the grid cells and must be separated by different feature channels.

Our solution is very simple: at the beginning of the network, we directly feed normalized pixel coordinates to the networks, inspired by 'CoordConv' operator [17]. Specifically, we create a tensor of same spatial size as input that

[3] We also show an equivalent and more efficient implementation in Section 4.

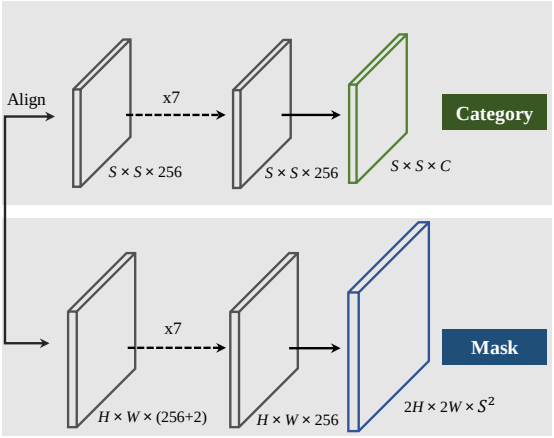


Fig. 3. SOLO Head architecture. At each FPN feature level, we attach two sibling sub-networks, one for instance category prediction (top) and one for instance mask segmentation (bottom). In the mask branch, we concatenate the x, y coordinates and the original features to encode spatial information. Here numbers denote spatial resolution and channels. In the figure, we assume 256 channels as an example. Arrows denote either convolution or interpolation. 'Align' means bilinear interpolation. During inference, the mask branch outputs are further upsampled to the original image size.

contains pixel coordinates, which are normalized to $[-1, 1]$. This tensor is then concatenated to the input features and passed to the following layers. By simply giving the convolution access to its own input coordinates, we add the spatial functionality to the conventional FCN model. It should be noted that CoordConv is not the only choice. For example the semi-convolutional operators [23] may be competent, but we employ CoordConv for its simplicity and being easy to implement. If the original feature tensor is of size $H \times W \times D$, the size of new tensor becomes $H \times W \times (D + 2)$, in which the last two channels are x-y pixel coordinates. For more information on CoordConv, we refer readers to [17].

Forming Instance Segmentation. In SOLO, the category prediction and the corresponding mask are naturally associated by their reference grid cell, *i.e.*, $k = i \cdot S + j$. Based on this, we can directly form the final instance segmentation result for each grid. The raw instance segmentation results are generated by gathering all grid results. Finally, non-maximum-suppression (NMS) is used to obtain the final instance segmentation results. No other post processing operations are needed.

2.2 Network Architecture

SOLO attaches to a convolutional backbone. We use FPN [14], which generates a pyramid of feature maps with different sizes with a fixed number of channels

(usually 256-d) for each level. These maps are used as input for each prediction head: semantic category and instance mask. Weights for the head are shared across different levels. Grid number may varies at different pyramids. Only the last conv is not shared in this scenario.

To demonstrate the generality and effectiveness of our approach, we instantiate SOLO with multiple architectures. The differences include: (a) the *backbone* architecture used for feature extraction, (b) the network *head* for computing the instance segmentation results, and (c) training *loss function* used to optimize the model. Most of the experiments are based on the *head* architecture as shown in Figure 3. We also utilize different variants to further study the generality. We note that our instance segmentation heads have a straightforward structure. More complex designs have the potential to improve performance but are not the focus of this work.

2.3 SOLO Learning

Label Assignment For category prediction branch, the network needs to give the object category probability for each of $S \times S$ grid. Specifically, grid (i, j) is considered as a positive sample if it falls into the *center region* of any ground truth mask, Otherwise it is a negative sample. Center sampling is effective in recent works of object detection [27,12], and here we also utilize a similar technique for mask category classification. Given the mass center (c_x, c_y), width w, and height h of the ground truth mask, the center region is controlled by constant scale factors ϵ: $(c_x, c_y, \epsilon w, \epsilon h)$. We set $\epsilon = 0.2$ and there are on average 3 positive samples for each ground truth mask.

Besides the label for instance category, we also have a binary segmentation mask for each positive sample. Since there are S^2 grids, we also have S^2 output masks for each image. For each positive samples, the corresponding target binary mask will be annotated. One may be concerned that the order of masks will impact the mask prediction branch, however, we show that the most simple row-major order works well for our method.

Loss Function We define our training loss function as follows:

$$L = L_{cate} + \lambda L_{mask}, \tag{1}$$

where L_{cate} is the conventional Focal Loss [15] for semantic category classification. L_{mask} is the loss for mask prediction:

$$L_{mask} = \frac{1}{N_{pos}} \sum_{k} \mathbb{1}_{\{\mathbf{p}^*_{i,j} > 0\}} d_{mask}(\mathbf{m}_k, \mathbf{m}^*_k), \tag{2}$$

Here indices $i = \lfloor k/S \rfloor, j = k \bmod S$, if we index the grid cells (instance category labels) from left to right and top to down. N_{pos} denotes the number of positive samples, $\mathbf{p}^*$ and $\mathbf{m}^*$ represent category and mask target respectively. $\mathbb{1}$ is the indicator function, being 1 if $\mathbf{p}^*_{i,j} > 0$ and 0 otherwise.

We have compared different implementations of $d_{mask}(\cdot,\cdot)$: Binary Cross Entropy (BCE), Focal Loss [15] and Dice Loss [21]. Finally, we employ Dice Loss for its effectiveness and stability in training. λ in Equation (1) is set to 3. The Dice Loss is defined as

$$L_{Dice} = 1 - D(\mathbf{p}, \mathbf{q}), \tag{3}$$

where D is the dice coefficient which is defined as

$$D(\mathbf{p}, \mathbf{q}) = \frac{2\sum_{x,y}(\mathbf{p}_{x,y} \cdot \mathbf{q}_{x,y})}{\sum_{x,y}\mathbf{p}_{x,y}^2 + \sum_{x,y}\mathbf{q}_{x,y}^2}. \tag{4}$$

Here $\mathbf{p}_{x,y}$ and $\mathbf{q}_{x,y}$ refer to the value of pixel located at (x, y) in predicted soft mask $\mathbf{p}$ and ground truth mask $\mathbf{q}$.

2.4 Inference

The inference of SOLO is very straightforward. Given an input image, we forward it through the backbone network and FPN, and obtain the category score $\mathbf{p}_{i,j}$ at grid (i, j) and the corresponding masks $\mathbf{m}_k$, where $k = i \cdot S + j$. We first use a confidence threshold of 0.1 to filter out predictions with low confidence. Then we select the top 500 scoring masks and feed them into the NMS operation. We use a threshold of 0.5 to convert predicted soft masks to binary masks.

Maskness. We calculate maskness for each predicted mask, which represents the quality and confidence of mask prediction maskness $= \frac{1}{\mathrm{N_f}}\sum_{\mathrm{i}}^{\mathrm{N_f}} \mathbf{p}_{\mathrm{i}}$. Here N_f the number of foreground pixels of the predicted soft mask $\mathbf{p}$, *i.e.*, the pixels that have values greater than threshold 0.5. The classification score for each prediction is multiplied by the maskness as the final confidence score.

3 Experiments

We present experimental results on the MS COCO instance segmentation benchmark [16], and report ablation studies by evaluating on the 5k `val2017` split. For our main results, we report COCO mask AP on the `test-dev` split, which has no public labels and is evaluated on the evaluation server.

Training details. SOLO is trained with stochastic gradient descent (SGD). We use synchronized SGD over 8 GPUs with a total of 16 images per mini-batch. Unless otherwise specified, all models are trained for 36 epochs with an initial learning rate of 0.01, which is then divided by 10 at 27th and again at 33th epoch. Weight decay of 0.0001 and momentum of 0.9 are used. All models are initialized from ImageNet pre-trained weights. We use scale jitter where the shorter image side is randomly sampled from 640 to 800 pixels, following [4].

3.1 Main Results

We compare SOLO to the state-of-the-art methods in instance segmentation on MS COCO `test-dev` in Table 1. SOLO with ResNet-101 achieves a mask AP

Table 1. Instance segmentation mask AP (%) on the COCO `test-dev`. All entries are *single-model* results. Here we adopt the "6×" schedule (72 epochs), following [4]. Mask R-CNN* is our improved version with scale augmentation and longer training time. D-SOLO means Decoupled SOLO as introduced in Section 4.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage:							
MNC [5]	Res-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [13]	Res-101-C5	29.2	49.5	–	7.1	31.3	50.0
Mask R-CNN [9]	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
MaskLab+ [3]	Res-101-C4	37.3	59.8	39.6	16.9	39.9	53.5
Mask R-CNN*	Res-101-FPN	37.8	59.8	40.7	20.5	40.4	49.3
one-stage:							
TensorMask [4]	Res-50-FPN	35.4	57.2	37.3	16.3	36.8	49.3
TensorMask [4]	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT [1]	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask [28]	Res-101-FPN	30.4	51.9	31.0	13.4	32.4	42.8
ours:							
SOLO	Res-50-FPN	36.8	58.6	39.0	15.9	39.5	52.1
SOLO	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
D-SOLO	Res-101-FPN	38.4	59.6	41.1	16.8	41.5	54.6
D-SOLO	Res-DCN-101-FPN	40.5	62.4	43.7	17.7	43.6	59.3

of 37.8%, the state of the art among existing *two-stage* instance segmentation methods such as Mask R-CNN. SOLO outperforms all previous *one-stage* methods, including TensorMask [4]. Some SOLO outputs are visualized in Figure 6, and more examples are in the supplementary.

3.2 How SOLO Works?

We show the network outputs generated by $S = 12$ grids (Figure 4). The subfigure (i, j) indicates the soft mask prediction results generated by the corresponding mask channel. Here we can see that different instances activates at different mask prediction channels. By explicitly segmenting instances at different positions, SOLO converts the instance segmentation problem into a position-aware classification task. Only one instance will be activated at each grid, and one instance may be predicted by multiple adjacent mask channels. During inference, we use NMS to suppress these redundant masks.

3.3 Ablation Experiments

Grid number. We compare the impacts of grid number on the performance with single output feature map as shown in Table 2. The feature is generated by merging C3, C4, and C5 outputs in ResNet (stride: 8). To our surprise, $S = 12$ can already achieve 27.2% AP on the challenging MS COCO dataset. SOLO achieves 29% AP when improving the grid number to 24. This results indicate

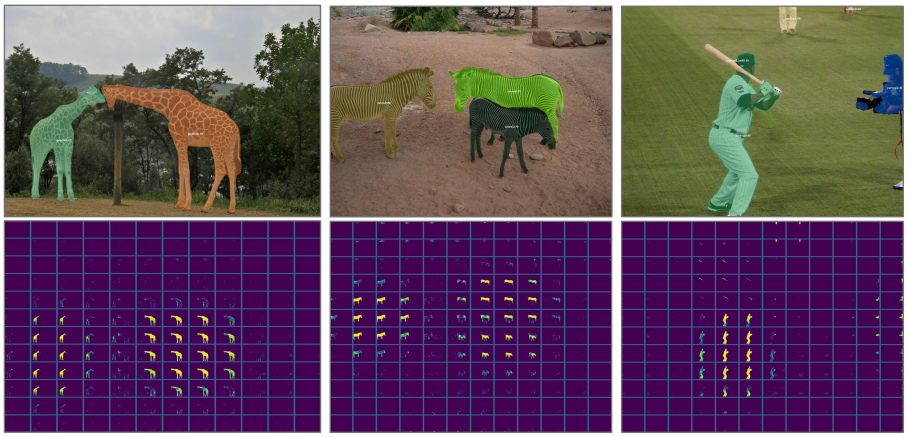


Fig. 4. SOLO behavior. We show the visualization of soft mask prediction. Here $S = 12$. For each column, the top one is the instance segmentation result, and the bottom one shows the mask activation maps. The sub-figure (i, j) in an activation map indicates the mask prediction results (after zooming out) generated by the corresponding mask channel.

Table 2. The impact of **grid number and FPN**. FPN significantly improves the performance thanks to its ability to deal with varying sizes of objects.

grid number	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
12	27.2	44.9	27.6	8.7	27.6	44.5
24	29.0	47.3	29.9	10.0	30.1	45.8
36	28.6	46.3	29.7	9.5	29.5	45.2
Pyramid	35.8	57.1	37.8	15.0	38.7	53.6

that our single-scale SOLO can be applicable to some scenarios where object scales do not vary much.

Multi-level Prediction. From Table 2 we can see that our single-scale SOLO could already get 29.0 AP on MS COCO dataset. In this ablation, we show that the performance could be further improved via multi-level prediction using FPN [14]. We use five pyramids to segment objects of different scales (details in supplementary). Scales of ground-truth masks are explicitly used to assign them to the levels of the pyramid. From P2 to P6, the corresponding grid numbers are $[40, 36, 24, 16, 12]$ respectively. Based on our multi-level prediction, we further achieve 35.8 AP. As expected, the performance over all the metrics has been largely improved.

CoordConv. Another important component that facilitates our SOLO paradigm is the *spatially variant* convolution (CoordConv [17]). As shown in Table 3, the standard convolution can already have spatial variant property to some extent, which is in accordance with the observation in [17]. As also revealed in [11], CNNs can implicitly learn the absolute position information from the commonly used

zero-padding operation. However, the implicitly learned position information is coarse and inaccurate. When making the convolution access to its own input coordinates through concatenating extra coordinate channels, our method enjoys 3.6 absolute AP gains. Two or more CoordConvs do not bring noticeable improvement. It suggests that a single CoordConv already enables the predictions to be well spatially variant/position sensitive.

Table 3. Conv vs. CoordConv. CoordConv can considerably improve AP upon standard convolution. Two or more layers of CoordConv are not necessary.

#CoordConv	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
0	32.2	52.6	33.7	11.5	34.3	51.6
1	35.8	57.1	37.8	15.0	38.7	53.6
2	35.7	57.0	37.7	14.9	38.7	53.3
3	35.8	57.4	37.7	15.7	39.0	53.0

Loss function. Table 4 compares different loss functions for our mask optimization branch. The methods include conventional Binary Cross Entropy (BCE), Focal Loss (FL), and Dice Loss (DL). To obtain improved performance, for Binary Cross Entropy we set a mask loss weight of 10 and a pixel weight of 2 for positive samples. The mask loss weight of Focal Loss is set to 20. As shown, the Focal Loss works much better than ordinary Binary Cross Entropy loss. It is because that the majority of pixels of an instance mask are in background, and the Focal Loss is designed to mitigate the sample imbalance problem by decreasing the loss of well-classified samples. However, the Dice Loss achieves the best results without the need of manually adjusting the loss hyper-parameters. Dice Loss views the pixels as a whole object and could establish the right balance between foreground and background pixels automatically. Note that with carefully tuning the balance hyper-parameters and introducing other training tricks, the results of Binary Cross Entropy and Focal Loss may be considerably improved. However the point here is that with the Dice Loss, training typically becomes much more stable and more likely to attain good results without using much heuristics. To make a fair comparison, we also show the results of Mask R-CNN with Dice loss in the supplementary, which performs worse (-0.9AP) than original BCE loss.

Alignment in the category branch. In the category prediction branch, we must match the convolutional features with spatial size $H \times W$ to $S \times S$. Here, we compare three common implementations: interpolation, adaptive-pool, and region-grid-interpolation. (a) Interpolation: directly bilinear interpolating to the target grid size; (b) Adaptive-pool: applying a 2D adaptive max-pool over $H \times W$ to $S \times S$; (c) Region-grid-interpolation: for each grid cell, we use bilinear interpolation conditioned on dense sample points, and aggregate the results with average. From our observation, there is no noticeable performance gap between these variants ($\pm$ 0.1AP), indicating that the alignment process does not have a significant impact on the final accuracy.

Table 4. Different loss functions may be employed in the mask branch. The Dice loss (DL) leads to best AP and is more stable to train.

mask loss	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
BCE	30.0	50.4	31.0	10.1	32.5	47.7
FL	31.6	51.1	33.3	9.9	34.9	49.8
DL	35.8	57.1	37.8	15.0	38.7	53.6

Different head depth. In SOLO, instance segmentation is formulated as a pixel-to-pixel task and we exploit the spatial layout of masks by using an FCN. In Table 5, we compare different head depth used in our work. Changing the head depth from 4 to 7 gives 1.2 AP gains. The results show that when the depth grows beyond 7, the performance becomes stable. In this paper, we use depth being 7 in other experiments.

Table 5. Different head depth. We use depth being 7 in other experiments, as the performance becomes stable when the depth grows beyond 7.

head depth	4	5	6	7	8
AP	34.6	35.2	35.5	35.8	35.8

Previous works (*e.g.*, Mask R-CNN) usually adopt four conv layers for mask prediction. In SOLO, the mask is conditioned on the spatial position and we simply attach the coordinate to the beginning of the head. The mask head must have enough representation power to learn such transformation. For the semantic category branch, the computational overhead is negligible since $S^2 \ll H \times W$.

3.4 SOLO-512

Speed-wise, the Res-101-FPN SOLO runs at 10.4 FPS on a V100 GPU (all post-processing included), vs. TensorMasks 2.6 FPS and Mask R-CNN's 11.1 FPS. We also train a smaller version of SOLO designed to speed up the inference. We use a model with smaller input resolution (shorter image size of 512 instead of 800). Other training and testing parameters are the same between SOLO-512 and SOLO.

Table 6. SOLO-512. SOLO-512 uses a model with smaller input size. All models are evaluated on `val2017`. Here the models are trained with "6×" schedule.

	backbone	AP	AP_{50}	AP_{75}	fps
SOLO	ResNet-50-FPN	36.0	57.5	38.0	12.1
SOLO	ResNet-101-FPN	37.1	58.7	39.4	10.4
SOLO-512	ResNet-50-FPN	34.2	55.9	36.0	22.5
SOLO-512	ResNet-101-FPN	35.0	57.1	37.0	19.2

With 34.2 mask AP, SOLO-512 achieves a model inference speed of 22.5 FPS, showing that SOLO has potentiality for real-time instance segmentation applications. The speed is reported on a single V100 GPU by averaging 5 runs.

3.5 Error Analysis

To quantitatively understand SOLO for mask prediction, we perform an error analysis by replacing the predicted masks with ground-truth values. For each predicted binary mask, we compute IoUs with ground-truth masks, and replace it with the most overlapping ground-truth mask. As reported in Table 7, if we replace the predicted masks with ground-truth masks, the AP increases to 68.1%. This experiment suggests that there are still ample room for improving the mask branch. We expect techniques developed (a) in semantic segmentation, and (b) for dealing occluded/tiny objects could be applied to boost the performance.

Table 7. Error analysis. Replacing the predicted instance mask with the ground-truth ones improves the mask AP from 37.1 to 68.1, suggesting that the mask branch still has ample room to be improved. The models are based on ResNet-101-FPN.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	37.1	58.7	39.4	16.0	41.1	54.2
w/gt mask	68.1	68.3	68.2	46.1	75.0	78.5

4 Decoupled SOLO

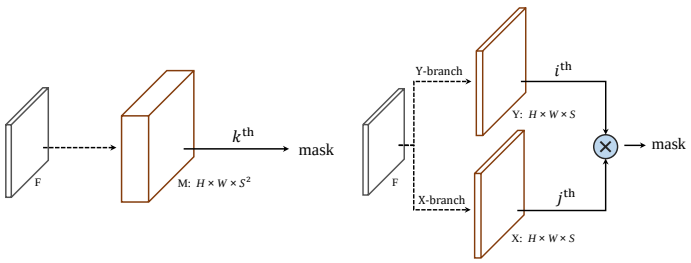
Given an predefined grid number, *e.g.*, $S = 20$, our SOLO head outputs $S^2 = 400$ channel maps. However, the prediction is somewhat redundant as in most cases the objects are located sparsely in the image. In this section, we further introduce an equivalent and significantly more efficient variant of the vanilla SOLO, termed **Decoupled SOLO**, shown in Figure 5.

In Decoupled SOLO, the original output tensor $M \in \mathbb{R}^{H \times W \times S^2}$ is replaced with two output tensors $X \in \mathbb{R}^{H \times W \times S}$ and $Y \in \mathbb{R}^{H \times W \times S}$, corresponding two axes respectively. Thus, the output space is decreased from $H \times W \times S^2$ to $H \times W \times 2S$. For an object located at grid location (i, j), the mask prediction of that object is defined as the element-wise multiplication of two channel maps:

$$\mathbf{m}_k = \mathbf{x}_j \otimes \mathbf{y}_i, \tag{5}$$

where $\mathbf{x}_j$ and $\mathbf{y}_i$ are the j^{th} and i^{th} channel map of X and Y after `sigmoid` operation. The motivation behind this is that the probability of a pixel belonging to location category (i, j) is the joint probability of belonging to i^{th} row and j^{th} column, as the horizontal and vertical location categories are independent.

We conduct experiments using the the same hyper-parameters as vanilla SOLO. As shown in Table 1, Decoupled SOLO even achieves slightly better



(a) Vanilla head (b) Decoupled head

Fig. 5. Decoupled SOLO head. F is input feature. Dashed arrows denote convolutions. $k = i \cdot S + j$. '$\otimes$' denotes element-wise multiplication.

performance (0.6 AP gains) than vanilla SOLO. With DCN-101 [6] backbone, we further achieve 40.5 AP, which is considerably better than current dominant approaches. It indicates that the Decoupled SOLO serves as an efficient and equivalent variant in accuracy of SOLO. Note that, as the output space is largely reduced, the Decoupled SOLO needs considerably less GPU memory during training and testing.

Fig. 6. Visualization of instance segmentation results using the Res-101-FPN backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 37.8 on the COCO `test-dev`.

5 Conclusion

In this work we have developed a direct instance segmentation framework, termed SOLO. Our SOLO is end-to-end trainable and can directly map a raw input im-

age to the desired instance masks with constant inference time, eliminating the need for the grouping post-processing as in bottom-up methods or the bounding-box detection and RoI operations in top-down approaches. Given the simplicity, flexibility, and strong performance of SOLO, we hope that our SOLO can serve as a cornerstone for many instance-level recognition tasks.

Acknowledgement We would like to thank Dongdong Yu and Enze Xie for the discussion about maskness and dice loss. We also thank Chong Xu and the ByteDance AI Lab team for technical support.

References

1. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: YOLACT: Real-time instance segmentation. In: Proc. IEEE Int. Conf. Comp. Vis. (2019)
2. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2019)
3. Chen, L.C., Hermans, A., Papandreou, G., Schroff, F., Wang, P., Adam, H.: Masklab: Instance segmentation by refining object detection with semantic and direction features. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2018)
4. Chen, X., Girshick, R., He, K., Dollar, P.: Tensormask: A foundation for dense object segmentation. In: Proc. IEEE Int. Conf. Comp. Vis. (2019)
5. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2016)
6. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proc. IEEE Int. Conf. Comp. Vis. (2017)
7. De Brabandere, B., Neven, D., Van Gool, L.: Semantic instance segmentation with a discriminative loss function. arXiv:1708.02551 (2017)
8. Gao, N., Shan, Y., Wang, Y., Zhao, X., Yu, Y., Yang, M., Huang, K.: Ssap: Single-shot instance segmentation with affinity pyramid. In: Proc. IEEE Int. Conf. Comp. Vis. (2019)
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: Proc. IEEE Int. Conf. Comp. Vis. (2017)
10. Huang, Z., Huang, L., Gong, Y., Huang, C., Wang, X.: Mask scoring R-CNN. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2019)
11. Islam*, M.A., Jia*, S., Bruce, N.D.B.: How much position information do convolutional neural networks encode? In: Proc. Int. Conf. Learn. Representations (2020)
12. Kong, T., Sun, F., Liu, H., Jiang, Y., Li, L., Shi, J.: Foveabox: Beyond anchor-based object detector. IEEE Trans. Image Process. pp. 7389–7398 (2020)
13. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2017)
14. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2017)
15. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proc. IEEE Int. Conf. Comp. Vis. (2017)
16. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Proc. Eur. Conf. Comp. Vis. (2014)

17. Liu, R., Lehman, J., Molino, P., Such, F.P., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. In: Proc. Advances in Neural Inf. Process. Syst. (2018)
18. Liu, S., Jia, J., Fidler, S., Urtasun, R.: Sequential grouping networks for instance segmentation. In: Proc. IEEE Int. Conf. Comp. Vis. (2017)
19. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2018)
20. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2015)
21. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: Proc. Int. Conf. 3D Vision (2016)
22. Newell, A., Huang, Z., Deng, J.: Associative embedding: End-to-end learning for joint detection and grouping. In: Proc. Advances in Neural Inf. Process. Syst. (2017)
23. Novotný, D., Albanie, S., Larlus, D., Vedaldi, A.: Semi-convolutional operators for instance segmentation. In: Proc. Eur. Conf. Comp. Vis. (2018)
24. Pinheiro, P.H.O., Collobert, R., Dollár, P.: Learning to segment object candidates. In: Proc. Advances in Neural Inf. Process. Syst. (2015)
25. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Proc. Advances in Neural Inf. Process. Syst. (2015)
26. Sofiiuk, K., Barinova, O., Konushin, A.: Adaptis: Adaptive instance selection network. In: Proc. IEEE Int. Conf. Comp. Vis. (2019)
27. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Proc. IEEE Int. Conf. Comp. Vis. (2019)
28. Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., Luo, P.: PolarMask: Single shot instance segmentation with polar representation. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. (2020)

A More Method Details

A.1 Multi-level Prediction

We use five FPN pyramids to segment objects of different scales (Table 8). Scales of ground-truth masks are explicitly used to assign them to the levels of the pyramid. Multi-level prediction gives 6.8 AP gains on the single-scale SOLO.

Table 8. we use five **FPN pyramids** to segment objects of different scales. The grid number increases for smaller instances due to larger existence space.

pyramid	P2	P3	P4	P5	P6
re-scaled stride	8	8	16	32	32
grid number	40	36	24	16	12
instance scale	<96	$48\sim192$	$96\sim384$	$192\sim768$	≥384

B More Experimental Results

B.1 Single-scale 1× Training

We list the 1x single-scale results in Table 9.

Table 9. Results with single-scale training. The models are trained with "1×" schedule and evaluated on `val2017`.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
SOLO	32.9	53.2	34.8	12.7	36.2	50.5
D-SOLO	33.9	54.0	35.7	13.8	36.9	51.0

B.2 Dice Loss

To make a fair comparison, we show the results of Mask R-CNN with Dice loss in Table 10. It shows that Dice loss is not suitable for Mask R-CNN, as it performs worse (-0.9AP) than original BCE loss. It is because the 'detect-then-segment' methods do not have the fg/bg imbalance issue, as they segment the foreground pixels in a local bounding box.

B.3 SOLO for Instance Contour Detection

Our framework can easily be extended to instance contour detection. We first convert the ground-truth masks in MS COCO into instance contours using OpenCV's `findContours` function, and then use the binary contours to optimize the mask branch in parallel with the semantic category branch. Here we

Table 10. Mask R-CNN with Dice loss. The models are trained with"3×" schedule and evaluated on `val2017`.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
BCE	36.2	58.0	38.9	20.1	39.5	49.0
DL	35.3	57.8	37.4	19.3	39.1	47.8

use Focal Loss to optimize the contour detection, other settings are the same with the instance segmentation baseline. Figure 7 shows some contour detection examples generated by our model. We provide these results as a proof of concept that SOLO can be used in contour detection.

Fig. 7. Visualization of SOLO for **instance contour detection**. The model is trained on COCO `train2017` dataset with ResNet-50-FPN. Each instance contour is shown in a different color.

B.4 Qualitative Results

We show more visualization results in Figure 8.

Fig. 8. Visualization of instance segmentation results using the Res-101-FPN backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 37.8 on the COCO `test-dev`.