Affinity Derivation and Graph Merge for Instance Segmentation

Yiding Liu[1(✉)], Siyu Yang[2], Bin Li[3], Wengang Zhou[1], Jizheng Xu[3], Houqiang Li[1], and Yan Lu[3]

[1] Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China
liuyd123@mail.ustc.edu.cn, {zhwg,lihq}@ustc.edu.cn
[2] Beihang University, Beijing, China
yangsiyu@buaa.edu.cn
[3] Microsoft Research, Beijing, China
{libin,jzxu,yanlu}@microsoft.com

Abstract. We present an instance segmentation scheme based on pixel affinity information, which is the relationship of two pixels belonging to the same instance. In our scheme, we use two neural networks with similar structures. One predicts the pixel level semantic score and the other is designed to derive pixel affinities. Regarding pixels as the vertexes and affinities as edges, we then propose a simple yet effective graph merge algorithm to cluster pixels into instances. Experiments show that our scheme generates fine grained instance masks. With Cityscape training data, the proposed scheme achieves 27.3 AP on test set.

Keywords: Instance segmentation · Pixel affinity · Graph merge
Proposal-free

1 Introduction

With the fast development of Convolutional Neural Networks (CNN), recent years have witnessed breakthroughs in various computer vision tasks. For example, CNN based methods have surpassed humans in image classification [24]. Rapid progress has been made in the areas of object detection [14,26,43], semantic segmentation [17], and even instance segmentation [19,21].

Semantic segmentation and instance segmentation try to label every pixel in images. Instance segmentation is more challenging as it also tells which object one pixel belongs to. Basically, there are two categories of methods for instance segmentation. The first one is developed from object detection. If one already has results of object detection, i.e. a bounding box for each object, one can move one step further to refine the bounding box semantic information to generate

This work was done when Yiding Liu and Siyu Yang took internship at Microsoft Research Asia.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11207, pp. 708–724, 2018.
https://doi.org/10.1007/978-3-030-01219-9_42

instance results. Since the results rely on the proposals from object detection, such a category can be regarded as proposal-based methods. The other one is to cluster pixels into instances based on semantic segmentation results. We refer to this category as proposal-free methods.

Recent instance segmentation methods have advanced in both directions above. Proposal-based methods are usually extensions of object detection frameworks [18,38,42]. Fully Convolutional Instance-aware Semantic Segmentation (FCIS) [33] produces position-sensitive feature maps [12] and generates masks by merging features in corresponding areas. Mask RCNN (Mask Region CNN) [23] extends Faster RCNN [44] with another branch to generate masks with different classes. Proposal-based methods produce instance-level results in the region of interest (ROI) to make the mask precise. Therefore, performance depends highly on the region proposal network (RPN) [44], and is usually influenced by the regression accuracy of the bounding box.

Meanwhile, methods without proposal generation have also been developed. The basic idea of these methods [4,15,28,34] is to learn instance level features for each pixel with a CNN, then a clustering method is applied to group the pixels together. Sequential group network (SGN) [37] uses CNN to generate features and makes group decisions based on a series of networks.

In this paper, we focus on proposal-free method and exploit semantic information from a new perspective. Similar to other proposal-free methods, we develop our scheme based on semantic segmentation. In addition to using pixel-wise classification results from semantic segmentation, we propose deriving pixel affinity information that tells if two pixels belong to the same object. We design networks to derive this information for neighboring pixels at various scales. We then take the set of pixels as vertexes and the pixel affinities as the weights of edges, constructing a graph from the output of the network. Then we propose a simple graph merge algorithm to group the pixels into instances. More details will be shown in Sect. 3.4. By doing so, we can achieve state-of-the-art results on the Cityscapes *test* set with only Cityscapes training data.

Our main contributions are as follows:

- We introduce a novel proposal-free instance segmentation scheme, where we use both semantic information and pixel affinity information to derive instance segmentation results.
- We show that even with a simple graph merge algorithm, we can outperform other methods, including proposal-based ones. It clearly shows that proposal-free methods can have comparable or even better performance than proposal-based methods. We hope that our findings will inspire more people to take instance segmentation to new levels along this direction.
- We show that a semantic segmentation network is reasonably suitable for pixel affinity prediction with only the meaning of the output changed.

2 Related Work

Our proposed method is based on CNNs for semantic segmentation, and we adapt this to generate pixel affinities. Thus, we first review previous works on semantic segmentation, followed by discussing the works on instance segmentation, which is further divided into proposal-based and proposal-free methods.

Semantic Segmentation: Replacing fully connected layers with convolution layers, Fully Convolutional Networks (FCN) [46] adapts a classification network for semantic segmentation. Following this, many works try to improve the network to overcome shortcomings [35,40,48]. To preserve spatial resolution and enlarge the corresponding respective field, [5,47] introduce dilated/atrous convolution to the network structure. To explore multi-scale information, PSPNet [48] designs a pyramid pooling structure [20,30,39] and Deeplabv2 [5] proposes Atrous Spatial Pyramid Pooling (ASPP) to embed contextual information. Most recently, Chen *et al.* have proposed Deeplabv3+ [8] by introducing an encoder-decoder structure [16,27,36,41] to [7] which achieves promising performance. In this paper, we do not focus on network structure design. Any CNN for semantic segmentation would be feasible for our work.

Proposal-Based Instance Segmentation: These methods exploit region proposals to locate the object and then obtain corresponding mask exploiting detection models [11,13,38,44]. DeepMask [42] proposes a network to classify whether the patch contains an object and then generates a mask. Multi-task Network Cascades (MNC) [10] provides a cascaded framework and decomposes instance segmentation into three phases including box localization, mask generation and classification. Instance-sensitive FCN [12] extends features to position-sensitive score maps, which contain necessary information for mask proposals, and generates instances combined with objectiveness scores. FCIS [33] takes position-sensitive maps further with inside/outside scores to encode information for instance segmentation. Mask-RCNN [23] adds another branch on top of Faster-RCNN [44] to predict mask outputs together with box prediction and classification, achieving excellent performance. MaskLab [6] combines Mask-RCNN with position-sensitive scores and improves performance.

Proposal-Free Instance Segmentation: These methods often consist of two branches, a segmentation branch and a clustering-purpose branch. Pixel-wise mask prediction is obtained by segmentation output and the clustering process aims to group the pixels that belong to a certain instance together. Liang *et al.* [34] predict the number of instances in an image and instance location for each pixel together with the semantic mask. They then perform a spectral clustering to group pixels. Long *et al.* [28] encode instance relationships to classes and exploit the boundary information when clustering pixels. Alireza *et al.* [15] and Bert *et al.* [4] try to learn the embedding vectors to cluster instances. SGN [37] tends to propose a sequential framework to group the instances gradually from points to lines and finally to instances, which currently achieves the best performance of proposal-free methods.

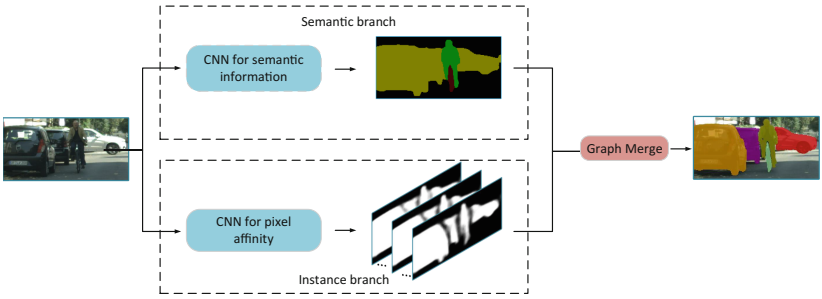

Fig. 1. Basic structure for the proposed framework.

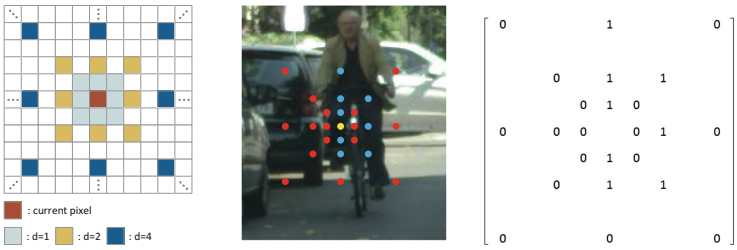

Fig. 2. Illustration for pixel affinity. (a) Locations for proposed neighbors. (b) The yellow point indicates the current pixel. Other points are neighboring pixels, in which red ones indicate pixels of different instances and blue ones indicate the same instance (*rider*). The pixel distance is NOT real but only for an illustration. (c) The derived labels and expected network output. (Color figure online)

3 Our Approach

3.1 Overview

The fundamental framework of our approach is shown in Fig. 1. We propose splitting the task of instance segmentation into two sequential steps. The first step utilizes CNN to obtain class information and pixel affinity of the input image, while the second step applies the graph merge algorithm on those results to generate the pixel-level masks for each instance.

In the first step, we utilize a semantic segmentation network to generate the class information for each pixel. Then, we use another network to generate information which is helpful for instance segmentation. It is not straightforward to make the network output pixel-level instance label directly, as labels of instances are exchangeable. Under this circumstance, we propose learning whether a pair of neighboring pixels belongs to the same instance. It is a binary classification problem that can be handled by the network.

It is impractical to generate affinities between each pixel and all the others in an image. Thus, we carefully select a set of neighboring pixels to generate

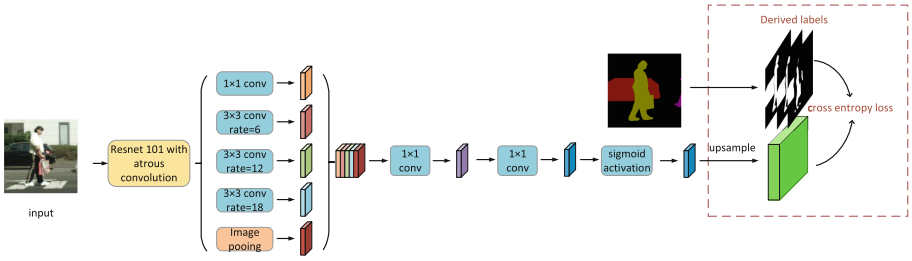

Fig. 3. Basic structure for instance branch, we utilize the basic framework from Deeplabv3 [7] based on Resnet-101 [25].

affinity information. Each channel of the network output represents a probability of whether the neighbor pixel and the current one belong to the same instance, as illustrated in Fig. 2(a). As can be seen from the instance branch in Fig. 1, the pixel affinities apparently indicate the boundary and show the feasibility to represent the instance information.

In the second step, we consider the whole image as a graph and apply the graph merge algorithm on the network output to generate instance segmentation results. For every instance, the class label is determined by voting among all pixels based on semantic labels.

3.2 Semantic Branch

Deeplabv3 [7] is one of the state-of-the-art networks in semantic segmentation. Thus, we use it as the semantic branch in our proposed framework. It should be noted that other semantic segmentation approaches could also be used in our framework.

3.3 Instance Branch

We select several pixel pairs, with the output of instance branch representing whether they belong to the same instance. Theoretically, if an instance is composed of only one connected area, we can merge the instance with only two pairs of pixel affinities, i.e whether $(p(x,y), p(x-1,y))$ and $(p(x,y), p(x,y-1))$ belong to the same instance, $p(x,y)$ is the pixel at location (x,y) in an image I. For the robustness to noise and ability to handle fragmented instances, we choose the following pixel set as the neighborhood of the current pixel $p(x,y)$

$$N(x,y) = \bigcup_{d \in D} N_d(x,y), \tag{1}$$

where $N_d(x,y)$ is the set of eight-neighbors of $p(x,y)$ with distance d, which can be expressed as

$$N_d(x,y) = \{p(x+a, y+b), \forall a, b \in \{d, 0, -d\}\} \setminus \{p(x,y)\}, \tag{2}$$

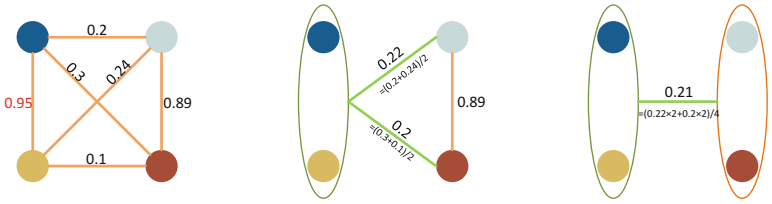

Fig. 4. A brief illustration for the graph merge algorithm

and D is the set of distances. In our implementation, $D = \{1, 2, 4, 8, 16, 32, 64\}$, as illustrated in Fig. 2(a).

We employ the network in Fig. 3 as the instance branch, in which we remove the last softmax activation of the semantic segmentation network and minimize the cross entropy loss after sigmoid activation. There are $8 \times 7 = 56$ elements in the set $N(x, y)$, so we assign 56 channels to the last layer. In the training procedure, the corresponding label is assigned as 1 if the pixel pair belongs to the same instance. In the inference procedure, we treat the network outputs as the probability of the pixel pair belonging to the same instance. We make a simple illustration of the selected neighbors in Fig. 2(b), and the corresponding label is shown in Fig. 2(c).

3.4 Graph Merge

The graph merge algorithm takes the semantic segmentation and pixel affinity results as input to generate instance segmentation results. Let vertex set V be the set of pixels and edge set E be the set of pixel affinities obtained from the network. Then, we have a graph $G = (V, E)$. It should be noted that the output of the instance branch is symmetrical. Pair $(p(x, y), p(x_c, y_c))$ obtained at (x, y) and $(p(x_c, y_c), p(x, y))$ at (x_c, y_c) have same physical meaning, both indicating the probability of these two pixels belonging to a certain instance. We average the corresponding probabilities before using them as the initial E. Thus, G can be considered an undirected graph (Fig. 4).

Let $e(i, j)$ denote an edge connecting vertex i and j. We first find the edge $e(u, v)$ with maximum probability and merge u, v together into a new super-pixel uv. It should be noted that we do not distinguish pixel and super-pixel explicitly, and uv is just a symbol indicating it is merged from u and v. After merging u, v, we need to update the graph G. For vertex set V, two pixels are removed and a new super-pixel is added,

$$V := V \setminus \{u, v\} \cup \{uv\}. \tag{3}$$

Then, the edge set E needs to be updated. We define $E(u) = \bigcup_{k \in K_u} \{e(u, k)\}$ representing all edges connecting with u. K_u is the set of pixels connecting to u. $E(u)$ and $E(v)$ should be discarded as u and v have been removed. $K_{uv} = K_u \cup K_v \setminus \{u, v\}$, E is updated as follows,

$$E := E \setminus E(u) \setminus E(v) \bigcup_{k \in K_{uv}} \{e(uv, k)\}. \tag{4}$$

For $k \in K(u) \cap K(v)$, $e(uv, k)$ is the average of $e(u, k)$ and $e(v, k)$. Otherwise, $e(uv, k)$ inherits from $e(u, k)$ or $e(v, k)$ directly.

After updating G, we continue to find a new maximum edge and repeat the procedure iteratively until the maximum probability is smaller than the threshold r_w. We summarize the procedure above in Algorithm 1. We then obtain a set of V and each pixel/super-pixel represents an instance. We recover the super-pixels to sets of pixels and filter the sets with a cardinality threshold r_c which means we only preserve the instance with pixels more than r_c. We get a set of pixels X as an instance and calculate the confidence of the instance from the initial E. We average all the edges $e(i, j)$ for both $i, j \in X$, and this confidence indicates the probability of X being an instance.

Algorithm 1. Graph Merge Algorithm

Require: Averaged instance branch output $P(u, v)$, thresholds r_w
Ensure: Merge result V, E
Initialize V with pixels and E with $e(u, v) = P(u, v)$
while Maximum $e(u, v) \in E \geq r_w$ **do**
  Merge u, v to super-pixel uv
  Update V: $V \Leftarrow V \setminus \{u, v\} \cup \{uv\}$
  $K_{uv} = K_u \cup K_v \setminus \{u, v\}$
  for $k \in K_{u,v}$ **do**
    if $k \in E(u) \cap E(v)$ **then**
      $e(uv, k)$ is the average of $e(u, k)$ and $e(v, k)$
    else
      $e(uv, k) = k \in K_u ?\ e(u, k) : e(v, k)$
    end if
  end for
  Update E: $E \Leftarrow E \setminus E(u) \setminus E(v) \bigcup_{k \in K_{uv}} \{e(uv, k)\}$
end while

We prefer the spatially neighboring pixels to be merged together. For this reason, we divide $D = \{1, 2, 4, 8, 16, 32, 64\}$ as three subsets $D_s = \{1, 2, 4\}$, $D_m = \{8, 16\}$ and $D_l = \{32, 64\}$ with which we do our graph merge sequentially. Firstly, we merge pixels with probabilities in D_s with a large threshold $r_{ws} = 0.97$, and then all edges with distances in D_m are added. We continue our graph merge with a lower threshold $r_{wm} = 0.7$ and repeat the operation for D_l with $r_{wl} = 0.3$.

4 Implementation Details

The fundamental framework of our approach has been introduced in the previous section. In this section, we elaborate on the implementation details.

4.1 Excluding Background

Background pixels do not need to be considered in the graph merge procedure, since they should not be present in any instance. Excluding them decreases the image size and accelerates the whole process. We refer to the interested sub-regions containing foreground objects as ROI in our method. Different from the ROI in the proposal-based method, the ROI in our method may contain multiple objects. In implementation, we look for connected areas of foreground pixels as ROIs. The foreground pixels will be aggregated to super-pixels when generating feasible areas for connecting the separated components belonging to a certain instance. In our implementation, the super-pixel is 32×32, which means if any pixel in a 32×32 region is foreground pixel, we consider the whole 32×32 region as foreground. We extend the connected area with a few pixels (16 in our implementation) and find the tightest bounding boxes, which is used as the input of our approach. Different from thousands of proposals used in the proposal-based instance segmentation algorithms, the number of ROIs in our approach is usually less than 10.

4.2 Pixel Affinity Refinement

Besides determining the instance class, the semantic segmentation results can help more with the graph merge algorithm. Intuitively, if two pixels have different semantic labels, they should not belong to a certain instance. Thus, we propose refining the pixel affinity output from the instance branch in Fig. 1 by scores from the semantic branch. We denote $P(x, y, c)$ as the probability of $p(x, y)$ and $p(x_c, y_c)$ belonging to a certain instance from the instance branch. We refine it by multiplying the semantic similarity of these two pixels.

Let $\mathbf{P}(x, y) = (p_0(x, y), p_1(x, y), \cdots, p_m(x, y))$ denote the probability output of the semantic branch. $m + 1$ denotes the number of classes (including background), $p_i(x, y)$ denotes the probability of the pixel belonging to the i-th class and $p_0(x, y)$ is the background probability. The inner product of the probabilities of two pixels indicates the probability of these two pixels having a certain semantic label. We do not care about background pixels, so we discard the background probability and calculate the inner product of $\mathbf{P}(x, y)$ and $\mathbf{P}(x_c, y_c)$ as $\sum_{i=1}^{m} p_i(x, y)p_i(x_c, y_c)$. We then refine the pixel affinity by

$$P_r(x, y, c) = \sigma(\sum_{i=1}^{m} p_i(x, y)p_i(x_c, y_c))P(x, y, c), \tag{5}$$

where

$$\sigma(x) = 2 \times (\frac{1}{1 + e^{-\alpha x}} - \frac{1}{2})_a. \tag{6}$$

This $\sigma()$ function is modified from the sigmoid function and we set $\alpha = 5$ to weaken the influence of the semantic inner product.

Fig. 5. Illustration for forcing local merge. We simulate the merging process with distance $\{1, 2, 4\}$ and window size 2, we only show edges involved in this process. Pixels with identical colors are merged and we update the new weights for edges. The left graph shows that the new probability in distance $\{2\}$ should only be averaged by the original weights from distance $\{4\}$ in the same direction. However, the right graph shows the new probability for distance $\{1\}$ should be an average of the edges from both distance $\{1\}$ and $\{2\}$.

Despite the information we mentioned above, we find that the semantic segmentation model may confuse different classes. Thus, we define the confusion matrix. The confusion matrix in semantic segmentation means a matrix where c_{ij} represents the count of pixels belonging to class i classified to class j. Given this, we can find that the semantic segmentation model sometimes misclassifies a pixel in a subclass, but rarely across sets. Thus, we combine classes in each set together as a super-class to further weaken the influence on instance segmentation from the semantic term. Moreover, we set the inner product to 0, when the two pixels are in different super-classes, which helps to refine the instance segmentation results.

4.3 Resizing ROIs

Like what ROI pooling does, we enlarge the shortened edge of the proposed boxes to a fixed size and proportionally enlarge the other edge, which we use as the input. For the Cityscapes dataset, we scale the height of each ROI to 513, if the original height is smaller than that. The reason of scaling it to 513 is that the networks are trained with 513×513 patches. Thus, we would like to use the same value for inference. Moreover, we limit the scaling factor to be less than 4. Resizing ROIs is helpful for finding more small instances.

4.4 Forcing Local Merge

We force the neighboring $m \times m$ pixels to be merged before the graph merge. During the process, we recalculate the pixel affinities according to our graph algorithm in Sect. 3.4. Figure 5 shows a simple example. Force merging neighboring pixels not only filters out the noises of the network output by averaging, but also decreases the input size of the graph merge algorithm to save on processing time. We will provide results on different merge window size in Sect. 5.3.

4.5 Semantic Class Partition

To get more exquisite ROIs, we refer to the semantic super-classes in Sect. 4.2 and apply it to the procedure of generating connected areas. We add together

the probabilities in each super-class and classify the pixels to super-classes. To find the foreground region of a super-class, we only consider the pixels classified to this super-class as foreground and all the others as background. Detailed experiment results will be provided in Sect. 5.3.

5 Experiment Evaluation

We evaluate our method on the Cityscapes dataset [9], which consists of 5,000 images representing complex urban street scenes with a resolution of 2048×1024. Images in the dataset are split into training, validation, and test sets of 2,975, 500, and 1,525 images, respectively. We use average precision (AP) as our metric to evaluate the results, which is calculated by the mean of the IOU threshold from 0.5 to 0.95 with the step of 0.05.

As most of the images in the Cityscapes dataset are background on top or bottom, we discard the parts with no semantic labeled pixels on the top or bottom for 90% of training images randomly, in order to make our data more effective. To improve semantic segmentation performance, we utilize coarse labeled training data by selecting patches containing *trunk, train,* and *bus* as additional training data to train the semantic branch. We crop 1554 patches from coarse labeled data. To augment data with different scale objects, we also crop several upsampled areas in the fine labeled data. As a result, the final patched fine labeled training data includes 14178 patches, including 2975 original training images with 90% of them having been dropped top and bottom background pixels. The networks are trained with Tensorflow [1] and the graph merge algorithm is implemented in C++.

5.1 Training Strategy

For the basic setting, the network output strides for both semantic and instance branch are set to 16, and they are trained with input images of size 513×513.

For the semantic branch, the network structure is defined as introduced in Sect. 3.2, whose weight is initialized with ImageNet [45] pretrained ResNet-101 model. During training, we use 4 Nvidia P40 GPUs with SGD [31] in the following steps. (1) We use 19-class semantic labeled data in the Cityscapes dataset fine and coarse data together, with an initial learning rate of 0.02 and a batch size of 16 per GPU. The model is trained using 100k iterations and the learning rate is multiplied by 0.7 every 15k iterations. (2) As the instance segmentation only focuses on 8 foreground objects, we then finetune the network with 9 classes labeled data (8 foreground objects and 1 background). Training data for this model contains a mix of 2 times fine labeled patched data and coarse labeled patches. We keep the other training setting unchanged. (3) We finetune the model with 3 times of original fine labeled data together with coarse labeled patches, with other training settings remaining unchanged.

For instance branch, we initialize the network with the ImageNet pretrained model. We train this model with patched fine labeled training data for 120k iterations, with other settings identical to step (1) in semantic model training.

Table 1. Instance segmentation performance on the Cityscapes *test* set. All results listed are trained only with Cityscapes.

Methods	person	rider	car	trunk	bus	train	mcycle	bicycle	AP 50%	AP
InstanceCut [29]	10.0	8.0	23.7	14.0	19.5	15.2	9.3	4.7	27.9	13.0
SAIS [22]	14.6	12.9	35.7	16.0	23.2	19.0	10.3	7.8	36.7	17.4
DWT [3]	15.5	14.1	31.5	22.5	27.0	22.9	13.9	8.0	35.3	19.4
DIN [2]	16.5	16.7	25.7	20.6	30.0	23.4	17.1	10.1	38.8	20.0
SGN [37]	21.8	20.1	39.4	**24.8**	33.2	**30.8**	17.7	12.4	44.9	25.0
Mask RCNN [23]	30.5	23.7	**46.9**	22.8	32.2	18.6	**19.1**	**16.0**	**49.9**	26.2
Ours	**31.5**	**25.2**	42.3	21.8	**37.2**	28.9	18.8	12.8	45.6	**27.3**

5.2 Main Results

As shown in Table 1, our method notably improves the performance and achieves 27.3 AP on the Cityscapes *test* set, which outperforms Mask RCNN trained with only Cityscapes *train* data by 1.1 points (4.2% relatively).

We show qualitive results for our algorithm in Fig. 6. As shown in the figure, we produce high quality results on both semantic and instance masks, where we get precise boundaries. As shown in the last row of results, we can handle the problem of fragmented instances and merge the separated parts together.

Our method outperforms Mask RCNN on AP but gets a relatively lower performance on AP 50%. This could mean we would get a higher score when the IOU threshold is larger. It also means that Mask RCNN could find more instances with relatively less accurate masks (higher AP 50%), but our method achieves more accurate boundaries. The bounding box of proposal-based method may lead to a rough mask, which will be judged as correct with a small IOU.

Utilizing the implementation of Mask RCNN in Detectron[1], we generate the instance masks and compare them with our results. As shown in Fig. 7, our results are finer grained. It can be expected that results will be better if we substitute the mask from Mask RCNN with ours when both approaches have prediction of a certain instance.

5.3 Detailed Results

We report the ablation studies with *val* set and discuss in detail.

Baseline: We take the algorithm we describe in Sect. 4.1 as the baseline, for excluding backgrounds helps to significantly speedup the graph merge algorithm and hardly affects the final results. We get 18.9% AP as our baseline, and we will introduce the results for strategies applied to the graph merge algorithm.

We show the experiment results for graph merge strategies in Table 2. For pixel affinity refinement, we add semantic information to refine the probability and get a 22.8% AP result. As shown in the table, it provides 3.9 points AP improvement. Then we resize the ROIs with a fixed size of 513, and we get a

[1] https://github.com/facebookresearch/Detectron.

Fig. 6. Results on Cityscapes *val* dataset, original image, semantic results, instance results and ground truth from left to right. Results in the last two rows are cropped from the original ones for better visualization.

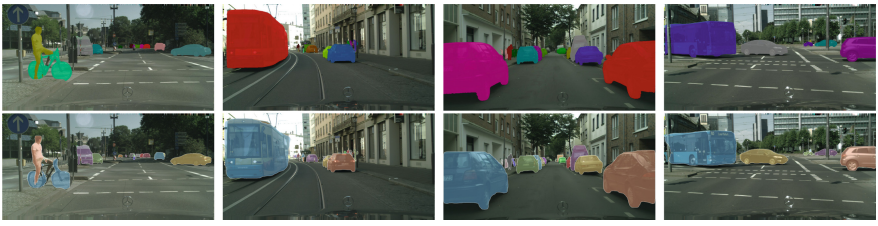

Fig. 7. Results compared with Mask RCNN. The first row is our results and the second row are results from Mask RCNN. As shown in the figure, we generate more fine-grained masks.

Fig. 8. Examples of faliure case

raise of 5.9 points AP, which significantly improve the results. The merge window size influences the results a lot. We have a 0.5 point improvement utilizing window size 2 and a 1.2 point drop with a window size of 4. As we can see, utilizing 2 as the window size not only reduces the complexity of graph merge, but also improves performance. Utilizing 4 causes a loss of detailed information and performs below expectations. Therefore, we utilize 2 in the following experiments. As mentioned in Sect. 4.1, we finally divide semantic classes into 3 subclasses for semantic class partition:{*person, rider*}, {*car, trunk, bus, train*} and {*motorcycle, bicycle*}, finding feasible areas separately. Such separation reduces

Table 2. Graph Merge Strategy: we test for our graph merge strategies for our algorithm including PAR: Pixel Affinity Refinement, RR: Resizing ROIs, FLM: Forcing Local Merge and SCP: Semantic Class Partition. Note that 2 and 4 in FLM represent the merge window size, default as 1.

PAR	RR	FLM	SCP	AP
				18.9
✓				22.8
✓	✓			28.7
✓	✓	2		29.2
✓	✓	4		27.5
✓	✓	2	✓	30.7

the influence across subclasses and makes the ROI resize more effectively. We get a 1.5 improvement by applying this technique from 29.0% to 30.5%, as shown in the table. It should be noted that utilizing larger images can make results better, but it also increases processing time.

Besides the strategies we utilize in the graph merge, we also test our model for different inference strategies referring to [7]. Output stride is always important for segmentation-like tasks. Small output stride usually means more detailed information but more inference time cost and smaller batch size in training. We test our models first trained on output stride 16, then we finetune models on output stride 8 as in [7]. It shows in Table 3 that both semantic and instance model finetuned with output stride 8 improve results by 0.5 point individually. When combined together, we achieve 32.1% AP with 1.4 point improvement compared with output stride 16.

We apply horizontal flips and semantic class refinement as alternative inference strategies. Horizontal flips for semantic inference brings 0.7 point increase in AP, and for instance inference flip, 0.5 point improvement is observed. We then achieve 33.5% AP combining these two flips.

Through observations on the *val* set, we find that instances in *bicycle* and *motorcycle* often fail to be connected when they are fragmented. To improve such situations, we map the pixel affinities between these two classes with Eq. 6 at the distance $d = 64$. As shown in Table 3, semantic class refinement yields 0.6 point improvement, and we get our best result of 34.1% AP on the *val* set.

5.4 Discussions

In our current implementation, the maximum distance of the instance branch output is 64. This means that the graph merge algorithm is not able to merge two non-adjacent parts with distance greater than 64. Adding more output channels can hardly help overall performance. Moreover, using other network structures, which could achieve better results on semantic segmentation may further improve the performance of the proposed graph merge algorithm. Some existing methods,

Table 3. Additional inference strategies: We test for additional inference strategies for our algorithm including Semantic OS: output stride for semantic branch, Instance OS: output stride for instance branch SHF: Semantic horizontal flip inference, IHF: Instance horizontal flip inference and SCR: Semantic Class Refinement. We also list several results from other methods for comparison.

Methods	Semantic OS	Instance OS	SHF	IHF	SCR	AP
DWT [3]						21.2
SGN [37]						29.2
Mask RCNN [23]						31.5
Ours	16	16				30.7
	8	16				31.2
	16	8				31.2
	8	8				32.1
	8	8	✓			32.8
	8	8		✓		32.6
	8	8	✓	✓		33.5
	8	8	✓	✓	✓	34.1

such as [32], could solve the graph merge problem but [32] is much slower than the proposed method. The current graph merge step is implemented on CPU and we believe there is big potential to use a multi-core CPU system for acceleration. Some examples of failure case are shown in Fig. 8. The proposed method may miss some small objects or merge different instances together by mistake.

6 Conclusions

In this paper, we introduce a proposal-free instance segmentation scheme via affinity derivation and graph merge. We generate semantic segmentation results and pixel affinities from two separate networks with a similar structure. Taking this information as input, we regard pixels as vertexes and pixel affinity information as edges to build a graph. The proposed graph merge algorithm is then used to cluster the pixels into instances. Our method outperforms Mask RCNN on the Cityscapes dataset by 1.1 point AP improvement using only Cityscapes training data. It shows that the proposal-free method can achieve state-of-the-art performance. We notice that the performance of semantic segmentation keeps improving with new methods, which can easily lead to performance improvement for instance segmentation via our method. The proposed graph merge algorithm is simple. We believe that more advanced algorithms can lead to even better performance. Improvements along these directions are left for further work.

Acknowledgement. Yiding Liu, Wengang Zhou and Houqiang Li's work was supported in part by 973 Program under Contract 2015CB351803, Natural Science Foundation of China (NSFC) under Contract 61390514 and Contract 61632019.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Arnab, A., Torr, P.H.S.: Pixelwise instance segmentation with a dynamically instantiated network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 879–888, July 2017. https://doi.org/10.1109/CVPR.2017.100
3. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2858–2866, July 2017. https://doi.org/10.1109/CVPR.2017.305
4. Brabandere, B.D., Neven, D., Gool, L.V.: Semantic instance segmentation for autonomous driving. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 478–480, July 2017. https://doi.org/10.1109/CVPRW.2017.66
5. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018). https://doi.org/10.1109/TPAMI.2017.2699184
6. 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. arXiv preprint arXiv:1712.04837 (2017)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
8. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611 (2018)
9. Cordts, M., et al.: The cityscapes dataset for semantic urban scene understanding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3213–3223, June 2016. https://doi.org/10.1109/CVPR.2016.350
10. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3150–3158, June 2016. https://doi.org/10.1109/CVPR.2016.343
11. Dai, J., et al.: Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 764–773, October 2017. https://doi.org/10.1109/ICCV.2017.89
12. Dai, J., He, K., Li, Y., Ren, S., Sun, J.: Instance-sensitive fully convolutional networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 534–549. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_32
13. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems, pp. 379–387 (2016)
14. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2155–2162, June 2014. https://doi.org/10.1109/CVPR.2014.276
15. Fathi, A., et al.: Semantic instance segmentation via deep metric learning. arXiv preprint arXiv:1703.10277 (2017)

16. Fu, J., Liu, J., Wang, Y., Lu, H.: Stacked deconvolutional network for semantic segmentation. arXiv preprint arXiv:1708.04943 (2017)
17. Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Garcia-Rodriguez, J.: A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv:1704.06857 (2017)
18. Girshick, R.: Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, December 2015. https://doi.org/10.1109/ICCV.2015.169
19. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, June 2014. https://doi.org/10.1109/CVPR.2014.81
20. Grauman, K., Darrell, T.: The pyramid match kernel: discriminative classification with sets of image features. In: Tenth IEEE International Conference on Computer Vision (ICCV 2005) Volume 1, vol. 2, pp. 1458–1465, October 2005. https://doi.org/10.1109/ICCV.2005.239
21. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Simultaneous detection and segmentation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 297–312. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_20
22. Hayder, Z., He, X., Salzmann, M.: Shape-aware instance segmentation. arXiv preprint arXiv:1612.03129 (2016)
23. He, K., Gkioxari, G., Dollr, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, October 2017. https://doi.org/10.1109/ICCV.2017.322
24. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034 (Dec 2015). https://doi.org/10.1109/ICCV.2015.123
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90
26. Huang, J., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3296–3297, July 2017. https://doi.org/10.1109/CVPR.2017.351
27. Islam, M.A., Rochan, M., Bruce, N.D.B., Wang, Y.: Gated feedback refinement network for dense image labeling. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4877–4885, July 2017. https://doi.org/10.1109/CVPR.2017.518
28. Jin, L., Chen, Z., Tu, Z.: Object detection free instance segmentation with labeling transformations. arXiv preprint arXiv:1611.08991 (2016)
29. Kirillov, A., Levinkov, E., Andres, B., Savchynskyy, B., Rother, C.: InstanceCut: from edges to instances with MultiCut. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7322–7331, July 2017. https://doi.org/10.1109/CVPR.2017.774
30. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 2, pp. 2169–2178 (2006). https://doi.org/10.1109/CVPR.2006.68
31. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989). https://doi.org/10.1162/neco.1989.1.4.541

32. Levinkov, E., et al.: Joint graph decomposition and node labeling: problem, algorithms, applications. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
33. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4438–4446, July 2017. https://doi.org/10.1109/CVPR.2017.472
34. Liang, X., Wei, Y., Shen, X., Yang, J., Lin, L., Yan, S.: Proposal-free network for instance-level object segmentation. arXiv preprint arXiv:1509.02636 (2015)
35. Lin, G., Shen, C., Van Den Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3194–3203, June 2016. https://doi.org/10.1109/CVPR.2016.348
36. Lin, T.Y., Dollr, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944 (July 2017). https://doi.org/10.1109/CVPR.2017.106
37. Liu, S., Jia, J., Fidler, S., Urtasun, R.: SGN: sequential grouping networks for instance segmentation. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 3516–3524, October 2017. https://doi.org/10.1109/ICCV.2017.378
38. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
39. Liu, W., Rabinovich, A., Berg, A.C.: ParseNet: looking wider to see better. arXiv preprint arXiv:1506.04579 (2015)
40. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1377–1385, December 2015. https://doi.org/10.1109/ICCV.2015.162
41. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 483–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_29
42. Pinheiro, P.O., Collobert, R., Dollár, P.: Learning to segment object candidates. In: Advances in Neural Information Processing Systems, pp. 1990–1998 (2015)
43. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, June 2016. https://doi.org/10.1109/CVPR.2016.91
44. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
45. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
46. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(4), 640–651 (2017). https://doi.org/10.1109/TPAMI.2016.2572683
47. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
48. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6230–6239, July 2017. https://doi.org/10.1109/CVPR.2017.660