S^{4}Net: Single Stage Salient-Instance Segmentation

Ruochen Fan[1] Qibin Hou[2] Ming-Ming Cheng[2] Tai-Jiang Mu[1] Shi-Min Hu[1]
[1]Tsinghua University [2]Nankai University
http://mmcheng.net/s4net/

Abstract

In this paper, we consider an interesting vision problem—salient instance segmentation. Other than producing approximate bounding boxes, our network also outputs high-quality instance-level segments. Taking into account the category-independent property of each target, we design a single stage salient instance segmentation framework, with a novel segmentation branch. Our new branch regards not only local context inside each detection window but also its surrounding context, enabling us to distinguish the instances in the same scope even with obstruction. Our network is end-to-end trainable and runs at a fast speed (40 fps when processing an image with resolution 320×320*). We evaluate our approach on a public available benchmark and show that it outperforms other alternative solutions. In addition, we also provide a thorough analysis of the design choices to help readers better understand the functions of each part in our network. To facilitate the development of this area, our code will be available at* https://github.com/RuochenFan/S4Net.

1. Introduction

Rather than recognizing all the objects in a scene, we humans only care about a small set of interesting objects/instances [29]. A recent experiment [13] demonstrates that interesting objects are normally visually salient, reflecting the importance of detection salient objects. In fact, localizing objects of interest is also essential for a wide range of computer graphics and computer vision applications. Such a capability allows many modern applications (*e.g.,* image manipulation/editing [7, 49, 5] and robotic perception [48]) to provide initial regions that might be of interest to users or robots so that they can directly proceed to image editing or scene understanding. Similar to [30], in this paper, we aim at detecting salient instances given an input image or photograph and simultaneously outputting their accurate instance-level segments.

Benefiting from the multi-level features extracted from convolutional neural networks (CNNs), recent object detection [17, 40, 11] methods provide better tools for localizing the bounding boxes of semantic objects. Additionally, there are also some other works [51] focusing on detecting approximate positions of salient objects (detection windows). These methods do provide various useful tools by finding salient objects, but they aim at providing bounding boxes instead of outputting instance-level object segments that are required for applications performing image editing [7, 5]. Recent instance-level semantic segmentation methods [9, 20] produce high-quality segmentations for each instance, however these works focus on semantic objects and hence are not competent to class-agnostic instance segmentation. Although it is possible for us to change the segmentation branches so that they can be applied to the binary case, these segmentation branches are based on either RoIWarp [9] or RoIAlign [20] layer, which only covers the feature information inside the bounding boxes. Compared to instance-level semantic segmentation, instance-level salient object segmentation, similar to salient object detection, focuses more on contrast information [26, 25]. This requires more *global context* [6] to be considered so as to highlight

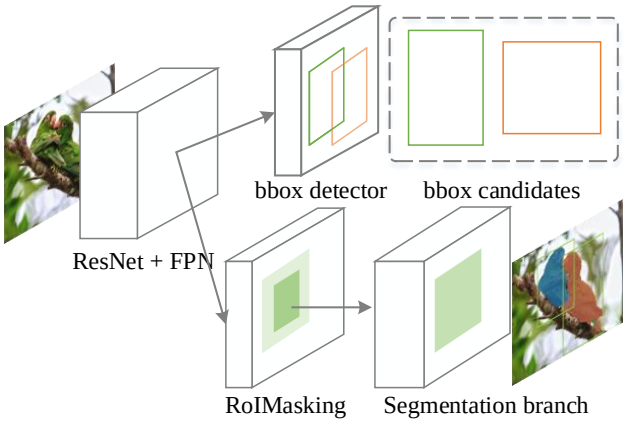


Figure 1: A brief illustration of the proposed method. In our segmentation branch, we propose a simple layer, namely RoIMasking, which takes into account both the information inside the bounding boxes and their surrounding context to better distinguish salient instances.

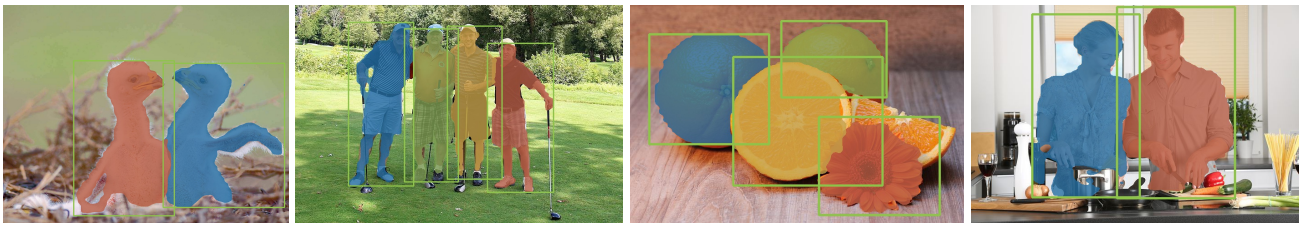


Figure 2: Illustrative examples produced by our approach. The input images are obtained from the Internet. Each instance is associated with a unique color in each image. We first use an object detector to localize the approximate windows of salient instances and then segment them out according to their surrounding context.

the instances of interest.

Taking the above line of thoughts into consideration, in this paper, we present a novel single-stage salient instance segmentation framework. Based on a single-stage object detector [34], we introduce a new segmentation branch aiming at producing pixel-wise segmentations for each salient instance. Unlike RoIPool [16, 21] and RoIAlign [20] which extract features only from limited scopes (bounding boxes), we propose a new region-based feature extraction layer, namely RoIMasking, to take into account the features inside the bounding boxes as well as more global information. Interestingly, RoIMasking is completely quantization-free and scale-preserving, allowing more detailed information to be successfully detected. Regarding the fact that salient instance segmentation relies more on global context, we design a new residual segmentation branch. This new branch not only substantially increases its own receptive field but also inherits the residual property of ResNets [22], leading to satisfactory performance. Beyond that, our model is end-to-end trainable and runs at 40fps on a single GPU when processing a 320×320 image.

To sum up, our proposed approach contains the following contributions:

- First, we propose an end-to-end single-shot salient instance segmentation framework. This model can not only achieve the state-of-the-art performance but also runs in real time.

- Second, we design a new RoIMasking layer which is able to preserve the original aspect ratio as well as resolution of the regions of interest, and meanwhile remain the context information around the regions of interest.

2. Related Works

Salient Object Detection. Salient object detection aims at jointly detecting the most distinguished objects and segmenting them out from a given scene. Early salient object detection methods mostly depended on either global or local contrast cues [6, 27]. They designed various hand-crafted features (e.g., color histogram and textures) for each region [2, 14, 43] and fused these features in either manual-designed or learning-based manners. Because of their weak ability of preserving the integrity of salient instances and the instability of hand-crafted features, these methods were gradually taken place by later CNN-based data-driven methods [23, 32, 47, 52, 15, 31, 30]. The key problems of these methods when applied to salient instance segmentation task are two-fold. First, the integrity of the salient objects are difficult to be preserved because the distinguished regions might be parts of the interesting instances. Second, salient object detection is a binary problem and hence cannot be competent to instance-level segmentation. Beyond only detecting salient objects, MSRNet [30] designed a series of mechanisms based on MCG [39] to provide instance-level salient object segmentation. Nevertheless, this method was excessively reliant on the quality of the edge maps and hence often failed when processing complicated real-world scenes.

Object Detection. The goal of object detection is to produce all the bounding box candidates for semantic categories. Earlier works mostly relied on hand-engineered features (*e.g.,* SIFT [37], SURF [4], and HOG [12]). They built different types of image pyramids so as to leverage more information across scales. Recently, the emergence of CNNs greatly promoted the development of object detectors. For example, R-CNN [17] and OverFeat [42] regarded CNNs as sliding window detectors for extracting high-level semantic information. Given a stack of pre-computed proposals [46, 8], these methods computed its feature vectors for each proposal using CNNs and then fed the features into a classifier. Later works [21, 16] took as inputs the entire images and applied region-based detectors to feature maps, substantially accelerating the running speed. Faster R-CNN [40] broke through the limitation of using pre-computed proposals by introducing a region proposal network (RPN) into CNNs. In this way, the whole network could be trained end-to-end, offering a better trade-off between accuracy and speed compared to previous works. However, all the method discussed above aim at outputting reliable object

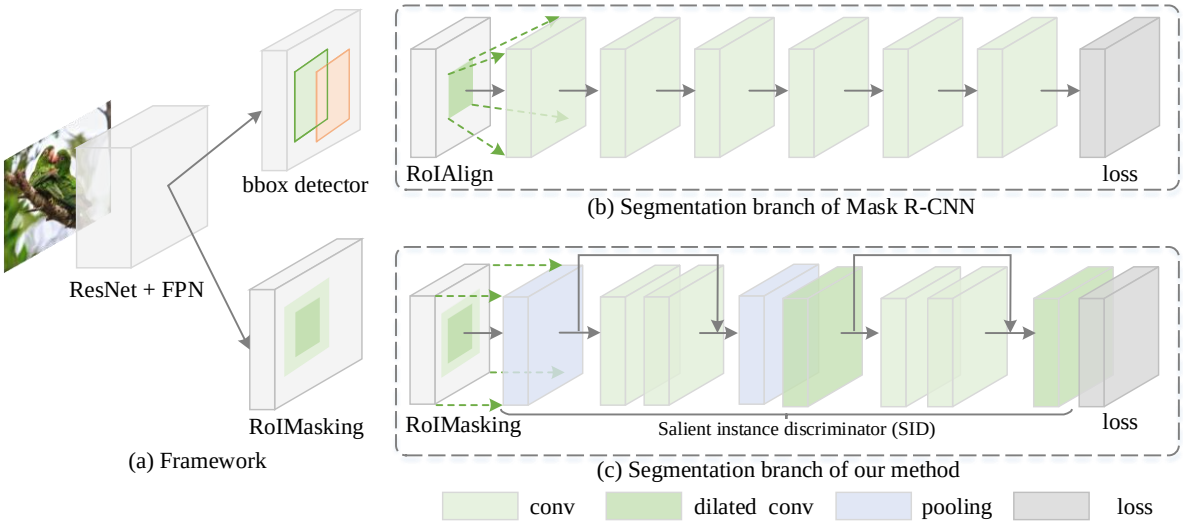


Figure 3: The overall architecture of the proposed method. (a) A brief illustration of our framework. (b) The segmentation branch proposed in [20] which is composed of a stack of consecutive convolutional layers. (c) Our proposed segmentation branch which further enlarges the size of receptive field but with the same number parameters as in (b).

proposals rather than instance segmentations.

Semantic Instance Segmentation. Earlier semantic instance segmentation methods [10, 18, 19, 38] were mostly based on segment proposals generated by segmentation methods [46, 39, 3]. In [9], Dai *et al.* predicted segmentation proposals by leveraging a multi-stage cascade to gradually refine rectangle regions from bounding box proposals. Li *et al.* [33] proposed to integrate the segment proposal network into an object detection network. More recently, He *et al.* implemented a Mask R-CNN framework, extending the Faster R-CNN [40] architecture by introducing a segmentation branch. Albeit more and more fascinating results, these methods are not suitable for our task as the segment proposals all belong to a pre-defined category collection. Sometimes the categories of interesting objects are unknown and thus our task requires class-agnostic segment proposals.

3. S^{4}Net

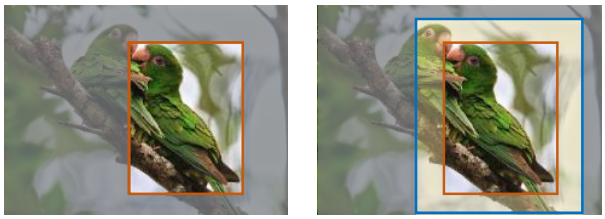
Regarding the demand of real time for most applications, we design a single-shot salient instance segmentation framework—*S^{4}Net*. S^{4}Net, as Mask R-CNN [20], introduces a new segmentation branch with RoIMasking to our single-shot object detector, which is easy to be implemented.

3.1. Framework

An overall architecture of S^{4}Net can be found in Fig. 3. Functionally, the framework of S^{4}Net can be separated into two components: a bounding box detector and a segmentation branch, both of which share the same base model as shown in Fig. 3. As in most object detection works, we select ResNet-50 [22] as our base model, which is pre-trained on the ImageNet dataset [41]. (Notice that we also exhibit the performance of other base models in our experiment section to verify the generality of the proposed framework.) For notational convenience, we divide ResNet-50 into 5 residual blocks, named conv1, ..., conv5 as in most works [34, 22].

The Backbone. As pointed out in most previous works [34], both local texture details and high-level abstract information are essential to detect objects, but for the base models themselves, it is difficult to contain both because of their bottom-up structure. Thus, inspired by the feature pyramid networks [34] and RON [28], in order to combine fine-grained details with highly-abstracted information, we adopt a U-shape hyper net structure. The output feature maps from conv2 to conv5 in ResNets are convoluted with convolutional layers with 1×1 kernel and 256 channels to make a series of intermediate lateral layers. A reverse connection can be made by upsampling an upper lateral layer and then combining it with a lower lateral layer by simple summation. Similar to [34], to avoid the alias effect brought in by summation, we add another convolutional layer with 3×3 kernel size and 256 channels after summation.

Single-Shot Object Detector. A detection branch is connected to the backbone to produce object bounding boxes. Considering the efficiency of the entire network, we adopt a single-shot object detector, which is similar to [35]. To leverage the multi-level features extracted from our U-shape backbone, we introduce 4 heads which are connected to each lateral layer. Each head structure is the same to the



(a) Binary masking (b) Ternary masking

Figure 4: Two different types of masks used in our RoIMasking layer. (a) Binary masking only considers the regions inside the orange rectangle, lacking of the ability of distinguishing salient instances. (b) Ternary masking takes into account both the region inside the orange rectangle and its surrounding regions marked in yellow.

one used in [40] but are with different strides in order to perform detection at multiple scales. Large objects are assigned to high-level feature maps with a big stride and more global information while small objects are assigned to low-level feature maps with a small stride and high resolution. It is worth mentioning that in single-shot detection model, negative samples are far more than positives samples. Unbalanced the positive and negative samples will greatly degrade the performance of the resulting proposals. Taking this into account, we calculate the objectness loss of positive and negative samples separately. In order to suppress false positives, online hard example mining (OHEM) [44] is used to calculate the objectness loss for negative samples.

Salient Instance Segmentation. The bounding boxes predicted by the detection branch and the output of the lateral layer with stride 8 in the backbone are fed into our RoIMasking layer, which will be described in detail thereafter. The RoIMasking layer marks out the regions of interest in feature maps and suppresses the information irrelevant to the interesting objects. In addition, a segmentation branch can be connected to the RoIMasking layer selectively. Taking the feature maps produced by RoIMasking layer as inputs, the segmentation branch outputs a series of saliency score maps by a fully-convolutional structure.

3.2. RoIMasking

RoIPool [16] and RoIAlign [20] are two standard fixed-size operations for extracting the features of the regions of interest. Both RoIPool and RoIAlign sample a region of interest into a fixed spatial extent of $H \times W$, and typically $H = W$. However, one of their drawbacks is that the sampling process is unable to maintain the original aspect ratio and resolution of the regions of interest. Besides, both RoIPool and RoIAlign focus on the regions inside the proposals, neglecting the rest area. In fact, the context around the regions of interest also makes evident sense to saliency segmentation but both RoIPool and RoIAlign discard it completely. In this subsection, we present a new resolution-preserving and quantization-free layer, called *RoIMasking*, to take the place of RoIPool or RoIAlign.

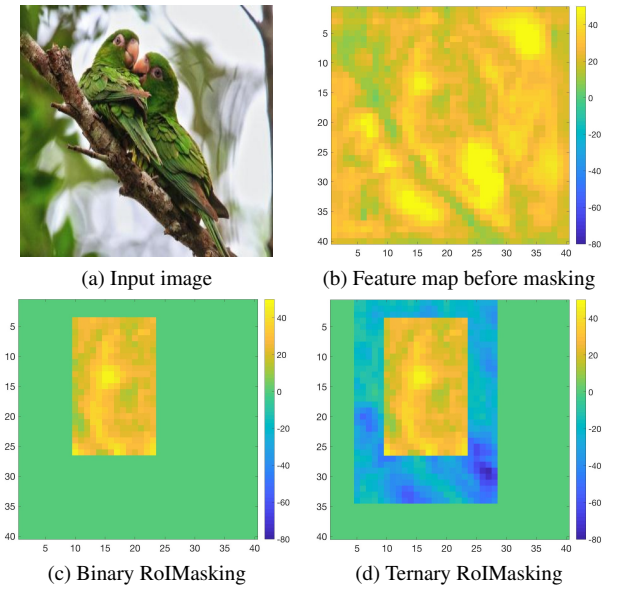
Binary RoIMasking We first introduce a simplified version of RoIMasking which we call binary RoIMasking. The binary RoIMasking receives feature maps and proposals predicted by the detection branch. A binary mask is generated according to the proposals, inside which the values are set to 1 and otherwise 0. Fig. 4a provides an illustration, in which the bright area is associated with label 1 and the dark region is with label 0. The output of the binary RoIMasking layer is the input feature maps multiplied by this mask. In Fig. 5, we show a typical example of the output feature maps. In the experiment part, we show that the proposed binary RoIMasking outperforms the RoIPool and RoIAlign baselines.

Ternary RoIMasking To make better use of the context information around the regions of interest, we further advance the binary RoIMasking to a ternary case. Because of the ReLU activation function, there is no negative value in the feature maps before RoIMasking. So, to better highlight the salient instances inside the proposals, we set the pixels around the regions of interest in the mask to -1. In Fig. 4b, the area between the blue and orange boxes is the corresponding area where pixels are set to -1. In this way, the features around regions of interest are distinct from those inside the bounding boxes of the salient instances. This allows the segmentation branch to be able to not only recognize which features belong to the regions of interest but also make use of the context information round the salient instances. The feature map after ternary RoIMasking is illustrated in Fig. 5d. It is worth mentioning that this operation introduces no more computational cost into our model. Ternary RoIMaking leads to a large improvement as we show in the experiment part. In the following, we abbreviate ternary RoIMasking as RoIMasking for notational convenience unless otherwise noted.

3.3. Segmentation Branch

Taking into account the structure of our backbone, we take the feature maps from the lateral layer associated with conv3 with a stride of 8 as the input to our segmentation branch on trade-off between global context and details. Before connecting our RoIMasking layer, we first add a simple convolutional layer with 256 channels and kernel size 1×1 for compressing the number of channels.

RoIMasking modifies feature maps by highlighting a series of areas bounding the instances and the corresponding perimeter context. However, it is still difficult for it to distinguish the salient instances from the other instances inside the same scope. To this end, we add a new module—*salient*



(a) Input image (b) Feature map before masking

(c) Binary RoIMasking (d) Ternary RoIMasking

Figure 5: The output feature maps of two different types of RoIMasking layers. (b) Before RoIMasking, all of the values in the feature map are non-negative because of the ReLU layer. (c) After binary RoIMasking, the regions outside the proposal is zeroed. (d) Ternary RoIMasking considers a larger area, which is depicted in blue.

instance discriminator (SID) to help better distinguish the instances even with large obstructions.

Having an overall look on the whole instance is crucial for distinguishing instances, so we should ensure the receptive field of SID is large enough. A detailed illustration of our SID module can be found in Fig. 3c. Other than two residual blocks, we also add two 3×3 max pooling with stride 1 and dilated convolutional layers with dilation rate 2 for enlarging the receptive field. All the convolutional layers has a kernel size 3×3 and stride 1. For the channel numbers, we set the first three to 128 and the rest 64, which we find are enough for salient instance segmentation.

3.4. Loss function

As described above, there are two sibling branches in our framework for detection and saliency segmentation, respectively. The detection branch undertakes objectness classification task and coordinates regression task, and the segmentation branch is for saliency segmentation task. Therefore, we use a multi-task loss L on each training sample to jointly train the model:

$$L = L_{obj} + L_{coord} + L_{seg}. \quad (1)$$

Regarding the fact that positive proposals are far less than negative samples in the detection branch, we adopt the following strategy. Let P and N be the collections of positive and negative proposals, N_P and N_N be the numbers of positive and negative proposals ($N_P \ll N_N$), then we calculate the positive and negative objectness loss separately to avoid the domination of negative gradients during training. Thus we have:

$$L_{obj} = -(\frac{1}{N_P}\sum_{i\in P}\log p_i + \frac{1}{N_N}\sum_{j\in N}\log(1-p_j)), \quad (2)$$

in which p_i is the probability of the ith proposal being positive.

$Smooth_{L1}$ loss is used for coordinate regression, and the overall regression loss is the mean of all $smooth_{L1}$ losses for the objects in an input image, which can be computed by

$$L_{coord} = \sum_i \frac{1}{N_{obj}} L^i_{coord}, \quad (3)$$

$$L^i_{coord} = \sum_{j\in\{x,y,w,h\}} smooth_{L1}(t^i_j - x^i_j), \quad (4)$$

in which t^i_j is the regression target for the coordinate x^i_j predicted by the detection branch for the ith object.

We also use cross-entropy loss for the segmentation branch. Despite the predicted score map has the same size with the feature map input to the segmentation branch, because of the RoIMasking, only parts of the score map are valid which corresponds to the field set to 1 in the mask. Let N_{valid} be the number of valid pixels in score map, the loss for segmentation branch is:

$$L_{seg} = -\frac{1}{N_{valid}}\sum_i (t_i \log p_i + (1-t_i)\log(1-p_i)), \quad (5)$$

in which p_i is the predicted probability of the pixel i belonging to foreground and t_i is 1 if pixel i belongs to foreground in ground truth, and 0 otherwise.

4. Experiments

In this section, we carry out detailed analysis to elaborate the functions of each component in our method by ablation studies. We also perform thorough comparisons with the state-of-the-art methods to exhibit the effectiveness of our approach. As salient instance segmentation is a much new vision problem with limited datasets released, we only use the dataset proposed in [30] for all experiments. This dataset contains 1,000 images with well-annotated instance-level annotations. For fair comparisons, as done in [30], we randomly select 500 images for training, 200 for validation, and 300 for testing.

4.1. Implementation Details

Training and Testing. In training phase, IoU is used to determine whether a bounding box proposal is a positive or

Methods	SID Module	$mAP^{0.5}$	$mAP^{0.7}$	$mAP^{0.5}_D$	$mAP^{0.7}_D$	$mAP^{0.5}_O$	$mAP^{0.7}_O$
RoIAlign [20]	✓	91.8%	67.1%	96.6%	76.6%	85.0%	53.7%
RoIPool [16]	✓	91.5%	65.3%	95.5%	72.5%	85.9%	55.3%
Binary RoIMasking	✓	91.4%	68.4%	95.0%	76.9%	86.3%	56.5%
Ternary RoIMasking	✗	90.9%	66.4%	96.0%	76.5%	83.9%	52.2%
Ternary RoIMasking	✓	**92.1%**	**69.6%**	95.4%	76.3%	87.5%	60.2%

Table 1: Quantitative results of the validation study. This table demonstrates that our proposed RoIMasking outperforms RoIAlign and RoIPool. Ternary RoIMasking and SID module bring performance gain mainly from the samples with obstructed samples.

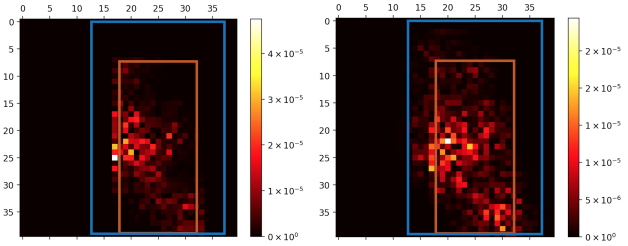
negative sample in detection branch. Recall that in Faster R-CNN, a bounding box proposal is assigned to be a positive sample if the IoU between the proposal and an object ground truth bounding box is more than 0.7. The proposal is assigned to be a negative sample if IoU is less than 0.3 and ignored between 0.3 and 0.7. However, we empirically found that using a single IoU threshold for reducing the false alarms is evidently better than the double IoU thresholds in Faster R-CNN. So in our detection branch, a bounding box proposal is positive if it's IoU > 0.5, and negative if IoU < 0.5.

In testing phase, the bounding boxes fed into RoIMasking layer are from the detection branch. But in training phase, we directly feed the ground truth bounding boxes into the RoIMasking layer. This provides the segmentation branch with more stable and valid training data and meanwhile accelerates the training process.

Hyper-parameters. Our proposed network is implemented on TensorFlow [1]. The input images are augmented by horizontal flipping. The hyper-parameters are set as follows: weight decay (0.0001) and momentum (0.9). We train our network on 2 GPUs for 20k iterations, with an initial learning rate of 0.004 which is divided by a factor of 10 at the 10k iteration. It only takes 40 minutes to train the whole model.

4.2. Analysis of RoIMasking

This subsection demonstrates the importance of the context information around the regions of interest in feature maps and the effectiveness of ternary RoIMasking. To do so, we explore the impact of each activations in the feature maps before RoIMasking on the performance. Inspired by [50], we visualize the function of a specific neuron in this model by drawing a gradients map. After loading the fully trained model weights, we do a forward pass using a specific image. In this process, the activation value of the feature maps before RoIMasking, $H_{i,j,c}$, is extracted and stored. Next, we do a backward pass. Note that in the general training stage, back-propagation is performed to calculate the gradients of the *total loss* with respect to the *weights* in neural network. But in this experiment, we load the stored $H_{i,j,c}$ as a variable, and regard the convolution



(a) Binary masking (b) Ternary masking

Figure 6: Gradient maps using binary masking and ternary masking. As can be seen, ternary masking considers more perimeter information of the orange proposal. The input image in this experiment is Figure 5a.

kernels as constant. Back-propagation is performed to calculate the gradients of the *saliency loss* with respect to each *feature map* input to RoIMasking:

$$G_{i,j,c} = \frac{\partial L_{sal}}{\partial H_{i,j,c}}. \tag{6}$$

The absolute value of $G_{i,j,c}$ reflects the importance of the feature map pixel $H_{i,j,c}$ to the saliency task. After summing up $|G_{i,j,c}|$ along the channel dimension, the gradient map $G_{i,j}$ can be obtained.

Fig. 6 shows the gradient maps for binary RoIMasking and ternary RoIMasking, respectively. The orange rectangle is the ground truth bounding box of a salient instance. By definition, the pixels inside the orange rectangle in the ternary mask are set to 0 and the pixels between the orange and blue boxes are set to -1. It is obvious that in Fig. 6b there are evident responses in the '-1' area. In Fig. 6a, there are only few responses between the orange and blue boxes. This phenomenon indirectly proves the importance of the context information around the regions of interest. More experimental results can be found in the subsequent parts.

4.3. Ablation Studies

In order to evaluate the effectiveness of each component in our proposed framework for instance-level salient object segmentation, we train our model on the salient in-

Figure 7: Selected examples of instance-level saliency segmentation results on the dataset proposed by [30] (the line above) and COCO [36] (the line below). Even obstructed instances can be well distinguished and segmented by our S^{4}Net.

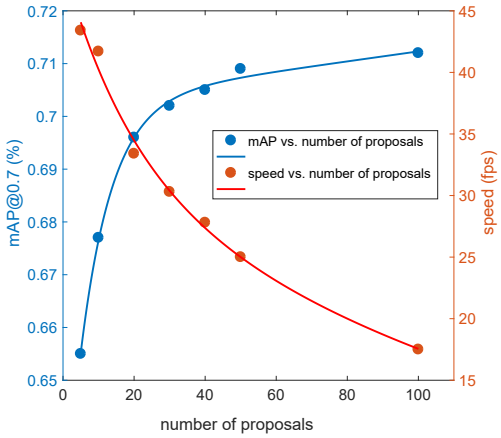


Figure 8: The performance and speed of S^{4}Net vs. the number of proposals. With the increase of proposals, better performance can be realized but the running time increases in parallel.

stance segmentation dataset by [30]. Following the standard COCO metrics [36], we report results on mAP (averaged precision over IoU thresholds), mAP$^{0.5}$, mAP$^{0.7}$, as well as mAP$_L$, mAP$_M$, mAP$_S$ for large, medium, and small instances. In order to analyze the ability to distinguish different instances, we divide the test set into two parts. One contains only separated instances, while the other comprises obstructed instances. Quantitative results for these two subsets are denoted by mAP$_D$ and mAP$_O$, respectively.

The Effect of RoIMasking. To evaluate the effectiveness of the proposed RoIMasking layer, we also consider using RoIPool and RoIAlign. We simply replace our RoIMasking with RoIPool and RoIAlign to perform two comparative experiments and keep other network structures and experimental conditions unchanged. Quantitative evaluation results are listed in Table 1. As can be seen, our proposed binary RoIMasking and ternary RoIMasking both outperform RoIPool and RoIAlign in mAP$^{0.7}$. Specifically, our ternary RoIMasking result improves the RoIAlign result by around 2.5 points. This reflects that considering more context information outside the proposals does help for salient instance segmentation.

The SID Module. An evaluation of our SID module is shown in Table 1. For this experiment, we only attempt to remove the SID module to show how much performance gain it brings in our framework. As can be seen on the right part of Table 1, the main difference between these two cases lies in the results of the images with obstructions. There is no evident performance gain on samples with only separated objects but a large improvement on images with obstructions (+8%). As a result, the major function of the SID module is to further distinguish different instances in the same scope of regions.

α	0	1/6	1/3	1/2	2/3	1
mAP$^{0.5}$	91.4%	92.0%	**92.1%**	92.0%	91.8%	91.7%
mAP$^{0.7}$	68.4%	69.4%	**69.6%**	69.3%	69.2%	69.0%

Table 2: Performance of S^{4}Net under different expansion coefficients. All the results shown here are based on ResNet-50 [22]. As can be observed, when $\alpha = 1/3$ we obtain the best result.

The Size of Context Regions. For better understanding our RoIMasking layer, we analyze how large the context regions should be here. Suppose bounding box size of a salient instance is (w, h). Here, we define an expansion coefficient α to denote the width of the '-1' region in the RoI mask. Hence, the size of the valid region is $(w + 2\alpha w, h + 2\alpha h)$. By default, we set α to 1/3. We

Methods	$mAP^{0.5}$	$mAP^{0.7}$	$mAP^{0.5}_D$	$mAP^{0.7}_D$	$mAP^{0.5}_O$	$mAP^{0.7}_O$	$mAP^{0.7}_L$	$mAP^{0.7}_M$	$mAP^{0.7}_S$
FCIS [33]	85.8%	62.5%	91.1%	71.8%	78.2%	49.5%	74.5%	63.2%	32.6%
MSRNet [30]	65.3%	52.3%	-	-	-	-	-	-	-
S^{4}Net	**92.1%**	**69.6%**	95.4%	76.3%	87.5%	60.2%	68.9%	73.6%	55.6%
S^{4}Net†	91.4%	68.4%	95.0%	76.9%	86.3%	56.5%	68.8%	73.3%	49.1%
S^{4}Net‡	90.9%	66.4%	96.0%	76.5%	83.9%	52.2%	68.4%	71.4%	45.1%

Table 3: Quantitative comparisons with existing methods on the 'test' set. '†' means S^{4}Net with binary RoIMasking. '‡' means S^{4}Net without SID module. As the instance segmentation maps of [30] and related code are not availabel, thus we use '-' to fill the blank cells.

Base models	mAP@0.5	mAP@0.7	Speed (FPS)
ResNet-101 [22]	**92.1%**	**72.4%**	33.3
ResNet-50 [22]	**92.1%**	69.6%	40.0
VGG16 [45]	87.4%	56.9%	43.5
MobileNet [24]	83.5%	49.1%	90.9

Table 4: Performance of S^{4}Net when using different base models. When we change the default ResNet-50 to ResNet-101, another 2.8% improvement can be obtained in spite of a little sacrifice on time cost. We also attempt to use the recent MobileNet [24] as our base model and yield a frame rate of more than 90 fps on a GTX 1080 Ti GPU.

also try different values of α to explore its influence on the final results as shown in Table 2 but found both larger and smaller values of α slightly harms the performance. This indicates that a region size of $(w+2w/3, h+2h/3)$ is enough for discriminating different instances.

The Number of Proposals. The number of proposals sent to the segmentation branch also has effect on the performance. According to our experiments, more proposals lead to better performance but more computational costs. Fig. 8 shows the relationship between performance and speed along with the increase of proposals. Notice that performance gain is not obvious when the number of proposals exceeds 20. Specially, when we set the number of proposals to 100, only around 1.5% improvement can be achieved but the running speed drops dramatically. Taking this into account, we take 20 proposals as a trade-off during the inference phase. Users may decide their own number of proposals in accordance with their tailored tasks.

Base Models. Besides the base model of ResNet-50 [22], we also try another three popular base models, including Resnet-101 [22], VGG16 [45], and MobileNet [24]. Table 4 lists the results when different base models are considered. As can be seen, base models with better performance on classification also works better in our experiments. For speed, real time processing can be achieved by our proposed S^{4}Net. When the size of input images is 320×320, S^{4}Net has a frame rate of 40.0 fps on a GTX 1080 Ti GPU. Furthermore, using MobileNet [24] as our base model, S^{4}Net runs very fast at a speed of 90.9 fps.

4.4. Comparisons with the State-of-the-Arts

As salient instance segmentation is a new problem, there is only one related work MSRNet [30] that can be used for comparison. For solidity, we train an instance-level semantic segmentation model FCIS [33] on the saliency dataset as an additional baseline. In this experiment, both our S^{4}Net and FCIS are based on ResNet-50 [22] pre-trained on ImageNet dataset. We report the results on the 'test' set.

Quantitative Analysis. The results of comparative experiments are listed in Table 3. Obviously, our proposed S^{4}Net achieves the best results in both $mAP^{0.5}$ and $mAP^{0.7}$. Specifically, our approach improve the baseline result presented in MSRNet [30] by about 27 points in $mAP^{0.5}$. In terms of $mAP^{0.7}$, we also have an improvement of more than 17 points on the same dataset. Compared to FCIS [33], our method wins by a large margin on each column of Table 3 as well. Even when only binary RoIMasking is used or SID module is removed, our approach still outperforms both MSRNet [30] and FCIS [33].

5. Conclusions

In this paper, we present the S^{4}Net, a single stage salient-instance segmentation framework, which is able to implement instance-level salient object segmentation in real time. Based on the single stage object detector, we introduce a novel segmentation branch, containing a novel RoIMasking layer and an advanced salient instance discriminator (SID). Our RoIMasking layer preserves the original resolution and aspect ratio of the regions of interest and at the meantime takes into account more context information outside the proposals. The SID module enlarges the receptive field of our segmentation branch, which further boosts the performance. Thorough experiments show that the proposed RoIMasking greatly outperforms RoIAlign and RoIPool, especially for distinguishing instances in the same scope. Our S^{4}Net achieves the state-of-the-art performance on a publicly available benchmark. Finally, we hope this framework can be carved a useful niche in image manipulation and robotic perception.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 6

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 2012. 2

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, 2011. 3

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). 2008. 2

[5] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: Internet image montage. *ACM TOG*, 28(5):124:1–10, 2009. 1

[6] M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S. Hu. Global contrast based salient region detection. *IEEE TPAMI*, 2015. 1, 2

[7] M.-M. Cheng, F.-L. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Repfinder: finding approximately repeated scene elements for image editing. In *ACM TOG*, volume 29, page 83, 2010. 1

[8] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014. 2

[9] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 1, 3

[10] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 3

[11] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[13] L. Elazary and L. Itti. Interesting objects are visually salient. *Journal of vision*, 2008. 1

[14] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 2

[15] L. Gayoung, T. Yu-Wing, and K. Junmo. Deep saliency with encoded low level distance map and high level features. In *CVPR*, 2016. 2

[16] R. Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015. 2, 4, 6

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 3

[19] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 3

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2, 3, 4, 6

[21] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI*, 2015. 2

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 3, 7, 8

[23] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. In *CVPR*, 2017. 2

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 8

[25] L. Itti and C. Koch. Computational modeling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001. 1

[26] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE TPAMI*, (11):1254–1259, 1998. 1

[27] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li. Salient object detection: A discriminative regional feature integration approach. In *CVPR*, pages 2083–2090, 2013. 2

[28] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen. Ron: Reverse connection with objectness prior networks for object detection. 2017. 3

[29] F.-F. Li, R. VanRullen, C. Koch, and P. Perona. Rapid natural scene categorization in the near absence of attention. *Proceedings of the National Academy of Sciences*, 2002. 1

[30] G. Li, Y. Xie, L. Lin, and Y. Yu. Instance-level salient object segmentation. In *CVPR*, 2017. 1, 2, 5, 6, 7, 8

[31] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *CVPR*, pages 5455–5463, 2015. 2

[32] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. 2

[33] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 3, 8

[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. 2017. 2, 3

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 3

[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 7

[37] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2

[38] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, 2015. 3

[39] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *IEEE TPAMI*, 2017. 2, 3

[40] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE TPAMI*, 2017. 1, 2, 3

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 3

[42] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2014. 2

[43] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 2000. 2

[44] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 4

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 8

[46] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2, 3

[47] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, pages 3183–3192, 2015. 2

[48] C. Wu, I. Lenz, and A. Saxena. Hierarchical semantic labeling for task-relevant rgb-d perception. In *Robotics: Science and systems*, 2014. 1

[49] H. Wu, Y.-S. Wang, K.-C. Feng, T.-T. Wong, T.-Y. Lee, and P.-A. Heng. Resizing by symmetry-summarization. In *ACM Transactions on Graphics (TOG)*, volume 29, page 159, 2010. 1

[50] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 6

[51] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech. Unconstrained salient object detection via proposal subset optimization. In *CVPR*, 2016. 1

[52] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *CVPR*, pages 1265–1274, 2015. 2