Fast Single Shot Instance Segmentation

Zuoxin Li[1], Fuqiang Zhou[1](✉), and Lu Yang[2]

[1] Beihang University, Beijing 100191, China
{lizuoxin,zfq}@buaa.edu.cn
[2] Beijing University of Posts and Telecommunication, Beijing 100876, China
soeaver@bupt.edu.cn

Abstract. In this work, we propose fast single shot instance segmentation framework (FSSI), which aims at jointly object detection, segmenting and distinguishing every individual instance (instance segmentation) in a flexible and fast way. In the pipeline of FSSI, the instance segmentation task is divided into three parallel sub-tasks: object detection, semantic segmentation, and direction prediction. The instance segmentation result is then generated from these three sub-tasks' results by the post-process in parallel. In order to accelerate the process, the SSD-like detection structure and two-path architecture which can generate more accurate segmentation prediction without heavy calculation burden are adopted. Our experiments on the PASCAL VOC and the MSCOCO datasets demonstrate the benefits of our approach, which accelerate the instance segmentation process with competitive result compared to MaskRCNN. Code is public available (https://github.com/lzx1413/FSSI).

Keywords: Instance segmentation · Multi-task learning · Convolutional Neural Networks

1 Introduction

Benefited from the Deep Convolutional Neural Networks (CNNs) [16], the field of category-level semantic segmentation and object detection has a rapid progress over a short period of time [3,23–25,29]. However, instance segmentation is more complex since it requires segmenting each instance in an image. To tackle this challenging task, several approaches have been proposed, such as MaskR-CNN [12] and FCIS [17]. Those methods can achieve state-of-the-art performance on the instance segmentation task but usually require a complicated pipeline and a huge amount of computations. Our goal in this work is to build a system which can split the instance segmentation task into several lightweight, parallel and computing sharing tasks. Hence they can run at a faster speed while retaining the competitive results with other methods.

There are mainly two kinds of methods to solve the instance segmentation problem. The first one is developing a system which focuses on bounding box detection first and then refining the prediction to obtain mask segmentation.

The state-of-the-art instance segmentation model MaskRCNN [12], generates the instance masks from the features cropped and resized by ROIAlign. The operator of ROIAlign is very heavy when there are numerous proposals. Besides, reducing the resolution of input images will decrease the performance markedly according to our experiments. Another type is to assign pixel predictions to instances in a bottom-up way. The prior work of [31] produces three predictions: semantic segmentation, instance direction (predicting the pixel's direction towards its corresponding instance center) and depth estimation. However, the complicate template matching which is used to decode the instance mask slows the speed of the pipeline.

In this work, we propose a fast single shot instance segmentation method (FSSI) which is a fast and flexible approach to tackle instance segmentation task. The instance segmentation task is decomposed into three parallel tasks: object detection, semantic segmentation, and direction prediction. In order to reduce the computational cost, the fusion feature generated from the base model is shared among the three tasks. We adopt the SSD-like head structure for object detection rather than crop-and-resize method introduced in [12,17]. Then the instance mask is produced by clustering the patch of masks from segmentation tasks according to the detected boxes, which is simpler and faster than template matching in the work [31]. As shown in Fig. 1, FSSI is designed in a multi-task way using one full convolutional network (FCN). Therefore, FSSI can be trained in an end-to-end way and the performance of instance segmentation can be further improved while better results can be produced by the three sub-tasks. Our main contributions can be summarized as follows:

(1) We decompose instance segmentation task into three sub-tasks: object detection, semantic segmentation, and direction prediction. We propose a novel and lightweight multi-task framework (FSSI) which shares the fusion feature and generates the outputs of the three sub-tasks.
(2) Our proposed FSSI adopts SSD-like object detection structure and two-path segmentation architecture to reduce the computational cost, which let FSSI perform about 4 times faster than MaskRCNN while having the competitive performance.
(3) We propose a simple and parallel post-process algorithm to generate instance segmentation masks from the three sub-task results.

2 Related Work

In this section, we summarize current instance segmentation algorithms based on deep neural networks and categorize them into two types, depending on the framework of the pipeline: Top-down or Bottom-up.

Top-Down Based Methods. Top-down based methods generate instance segmentation by producing the bounding box of objects firstly. Then the regions of features in the bounding boxes are cropped and resized to be used to either classify mask regions or refine boxes to obtain instance masks. There have been several methods proposed to tackle the instance segmentation task. MCG [1], SharpMask [28] and instance-sensitive FCNs [4] are proposed to generate mask proposals. MNC [5] decomposes the instance segmentation into three cascade problems including box localization, mask refinement, and instance classification. The former result is used as the input of the following tasks. Hayer*et al.* [9] suggest reducing the mask boundary error to improve MNC. Higher-order Conditional Random Fields (CRFs) are used in [2] to refine the instance mask. FCIS [17] is the first Fully Convolutional Network (FCN) for instance segmentation. It adopts position-sensitive score maps from InstanceFCN [4] and considers inside/outside score maps further to produce instance masks. Recently, Mask-RCNN [12], which is built on the top of FPN [20], adopts another small FCN branch to obtain refined mask results from box predictions.

Bottom-Up Based Methods. Bottom-up based methods generally adopt the following two stages, including segmentation and clustering. The segmentation module predicts the pixel-level labels, and the clustering process is used to group the per-pixel predictions together for each object instance. PFN [19] adopts spectral clustering to group the semantic segmentation results from DeepLab [3] depending on the prediction of object number and the bounding box every pixel belongs to. Zhang *et al.* [34] apply depth order to distinguish different instances. Uhrig *et al.* [31] train an FCN to predict the direction to the object center of each pixel, semantic mask, and depth information to generate instance segmentations. Liu *et al.* [22] segment objects from patches of images with multi-scales and aggregate them together. However, existing methods seldom pay attention to the efficiency of instance segmentation and they are hard to be deployed to platforms which have limited computational capability. For instance, MaskRCNN based on ResNet-50 [13] can only run at 5 FPS on the advanced GPUs. To tackle this problem, our proposed FSSI model combines the advantages from both top-down and bottom-up advantages and aims at supplying a fast and lightweight approach to generate instance masks. Our proposed method divides the instance segmentation task into three parallel sub-tasks: object detection, semantic segmentation, and direction segmentation. Different from FCIS or MaskRCNN, which are built on the two-stage object directors and refine the mask from the features w.r.t the detected boxes, our FSSI adopts SSD-like head for object detection and the results of instance segmentation initialized by the boxes are generated from the semantic segmentation and direction prediction directly. Benefited from the two-path architecture described in Sect. 3, the computational cost for segmentation tasks is reduced observably.

3 Fast Single Shot Instance Segmentation

In this section, we first introduce the sub-tasks designed to accomplish the final instance segmentation task. Then we demonstrate the global view of our FSSI pipeline and the architecture of each sub-task. Finally, we explain how to aggregate results from the sub-tasks to generate the instance segmentation result and the loss functions which are used to optimize our FSSI.

3.1 Multi-task Design for Instance Segmentation

In our FSSI framework, instance segmentation can be decomposed into three parallel tasks including object detection, semantic segmentation and auxiliary information for distinguishing individual instances which belong to the same category. The object detection result can supply the category label and bounding box information which can be regarded as the initial region of interest to generate a refined instance mask. Semantic segmentation logits are designed to predict the pixel-wise semantic logits, which have the ability to separate instances of different semantic labels. However, semantic labels are not sufficient to make a distinction between instance objects which belong to the same class and are connected to each other. Auxiliary information for distinguishing them is necessary at this time. There are several ways to tackle this task, such as direction, boundary, and depth prediction. In this work, we adopt the direction prediction proposed by [31] as the auxiliary information. The direction prediction logits can be used to predict each pixel's direction towards of the center of its corresponding instance. However, our direction target is different from [31], which will be discussed in Sect. 3.6. The instance segmentation results can be obtained by a simple post-processing method which will be described in Sect. 3.7 in details.

3.2 Global View of the Pipeline

As presented in Fig. 1, we design one fully convolutional network, which performs object detection, semantic segmentation, and direction prediction synchronously. Our motivation is as follows: Firstly, we consider generating a comprehensive representation of the input image. Then the general feature will be regarded as the input feature for three subnets which are corresponding to the three sub-tasks. Finally, outputs of the three sub-tasks will be aggregated together to generate the instance segmentation results with a post-process module.

3.3 Fusion Feature

Inspired by HyperNet [15] and FSSD [18], we produce the fusion feature by fusing the multi-level feature maps generated by the base model. Considering the trade-off between the performance and speed in image classification task, we choose ResNet-50 [13] as the base model. The fusion feature is the comprehensive representation of the input image. Following FSSD [18], we choose the feature maps with feature stride 8, 16, 32 to generate the fusion feature, whose feature stride is 8. In order to reduce the time consumed in the fusion feature extraction, we resize the input image to 300×300 before feeding it into the base model.

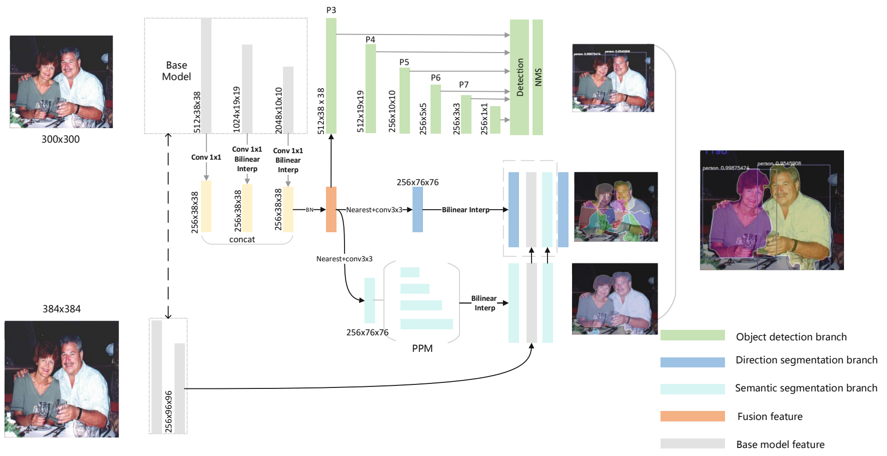

Fig. 1. The FSSI architecture. The size of the input image is 384 × 384. FSSI consists of three branches for **object detection**, **semantic segmentation,** and **direction prediction**. We adopt ResNet50 [13] as the **BaseModel** to extract the **fusion feature**(orange). The green part is the single shot object detection sub-net which takes the **fusion feature** as the input. As for the segmentation tasks, the original large input image is used to generate **low level feature** by a shallow FCN (shares weights with part of the base model). Then one pyramid pooling module (PPM) is appended after the fusion feature to generate high-level semantic feature map. Next, the low-level and high-level feature maps are concatenated together to generate the semantic segmentation results, which will be concatenated with the feature maps from the direction segmentation sub-net to produce the direction prediction outputs. Finally, the instance segmentation results can be generated by a post-process.

3.4 SSD Head for Object Detection

SSD [23] is an excellent object detector architecture which can achieve outstanding performance while running at a very fast speed. Inspired by FSSD [18], we generate new feature pyramid from the fusion feature with a cascade of convolutional layers whose stride is 2. Then the feature pyramid will be adopted to generate classification and coordinate correction for a set of default bounding boxes. At test time, non-maximum suppression (NMS) will be used to filter the final object detection results. Since we only use the image with 300 × 300 to process object detection, the object detection process is very efficient.

3.5 Segmentation Sub-networks

As shown in Fig. 1, we append two sub-nets after the fusion feature to generate prediction map for the semantic mask and direction mask. Since the spatial size of the fusion feature is 38 × 38, it is insufficient to generate the mask with high quality, especially for the direction map whose spatial size is much smaller than the semantic mask (semantic masks are divided into direction masks). Therefore, we adopt the two-path structure, which means that we also extract the low-level feature maps from the original input image (384×384) using a sequence of convolutional layers which share the weights with the first stage of ResNet-50 to refine the mask prediction results.

Semantic Segmentation Branch. As the feature stride of the fusion feature is 8, we firstly up-sample the feature map by the nearest interpolation layer with up-scale factor 2 followed by one convolutional layer. We also leverage the pyramid pooling module (PPM) which is proposed by PSPnet [35] to enhance the capacity of the semantic segmentation branch. The high-level feature maps generated from the fusion feature will be concatenated to the low-level feature maps to generate the ultimate semantic segmentation results. The output semantic prediction map will be up-sampled by bilinear interpolation to match the size of the input image.

Direction Segmentation Branch. The target of direction segmentation is described in Sec. 3.6. Compared with semantic segmentation target, the direction segmentation target has a smaller area. Therefore, we discard the PPM and only up-sample the fusion feature by a nearest up-sample layer with scale factor 2 and a convolutional layer. Since the direction segmentation only needs to be conducted on the region of objects, the output of the semantic segmentation is concatenated with the up-sampled fusion feature to serve as a reference for the objects' spatial distribution. However, the gradient of the direction segmentation branch does not be back-propagated to the semantic segmentation branch. Different from the FCIS [17], our proposed direction results are class-agnostic instead of having the logits for each category to yield more compact models. Hence the outputs of direction segmentation are only 9 channels (8 for 8 directions and 1 for the background.)

3.6 Direction Map of Objects

As shown in Fig. 2(b), semantic segmentation will only predict the category results of the pixels rather than instance results. In order to discriminate the different object instances which belong to the same category and are connected together in the image, we adopt the direction map (Fig. 2(c)) which is inspired by [31]. [31] splits the instance mask into 8 different directions with the angle interval of 45 degrees. Different from the [31], we use the diagonal and the lines between the midpoints on the sides of the bounding boxes to split the instance mask into 8 regions (We do not adopt 4 or 16 regions because they do not perfrom as well as 8 regions in our experiments.). We propose this method according to fact that the input image will be resized before being processed by the network. Our approach can preserve the constant direction label with image-ratio varieties. Besides, this strategy can also make the direction logits become easier to collaborate with the object detection results, which will be described in Sect. 3.7. Besides, since the area of each individual direction segmentation mask is smaller than the original category mask and only the regions where objects exist need to generate direction

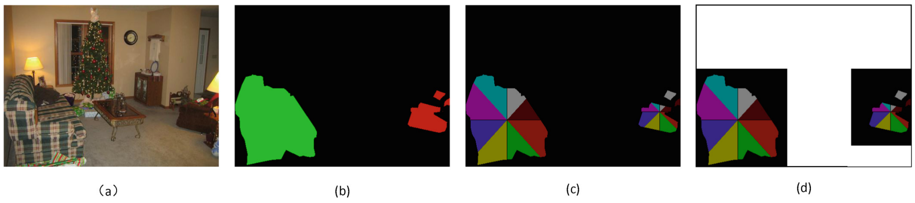

Fig. 2. Segmentation targets. (a) input image. (b) semantic segmentation target. (c) original direction segmentation target. (d) direction segmentation target after ignoring the background regions (in white color).

predictions, we ignore the region (Fig. 2(d)) which is outside of the expanded bounding boxes from the ground truths to reduce the jamming effect from the background.

3.7 Post-process for Generating Instance Mask

The multi-task network called FSSI can generate the object detection, semantic segmentation and direction segmentation of the input image. The post-process will take these three parts of information as the inputs and generate the instance segmentation results. For the objects in one image, we process them with groups which are clustered by their category labels from the object detection results. Different groups of objects can be processed in parallel to accelerate the process. For each group of objects, as shown in Fig. 3, there are two stages for post-process.

1. Find and match the bounding boxes (det-box) with the individual semantic masks, which means that the masks are not connected with other objects in the same group. First, connected components of the semantic masks are extracted and the bounding box of these connected components (mask-box) can be calculated. Then the intersection over union (IoU) between each det-box and mask-box can be obtained. One det-box and one mask-box are matched successfully if the IoU between them is higher than a threshold t_{bs} which is 0.7 in our experiments. Therefore, we can confirm one object with both it's bounding box and mask. In order to avoid the problem that the det-box is not large enough to surround the object (last row in Fig. 4), which leads to failure to produce the full mask of the object, we set the union set of mask-box and det-box to be the final object box in this stage.
2. Discriminate the instance masks which are connected together with the guide of direction predictions. First, given one predicted bounding box and category label, we crop the regions in the semantic segmentation map and direction map w.r.t the predicted bounding box. The semantic patch is transformed into a binary mask according to the category label and used to perform as the mask to filter the direction patches. Regions in the direction map which are not in the semantic mask will turn to be background. Then the direction

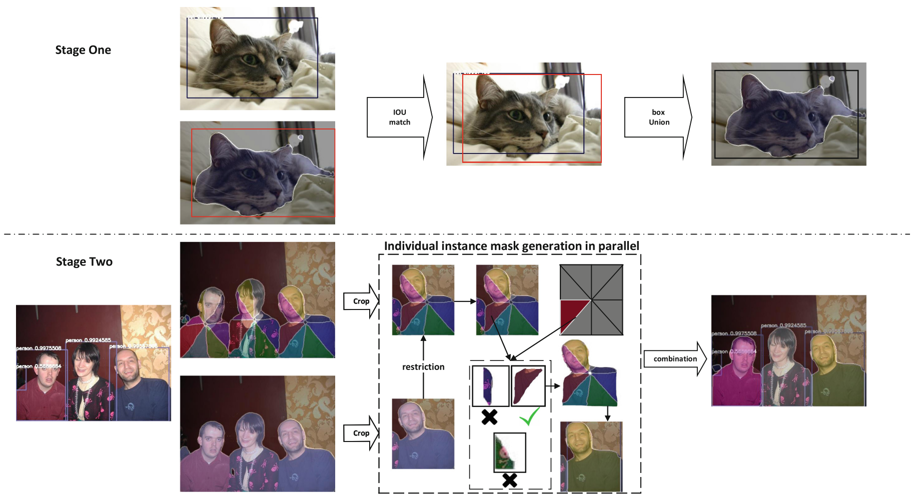

Fig. 3. Pipeline of the post-process for instance results. Details can be found in Sect. 3.7.

patch can be used to determine the mask of center objects according to their location distribution as follows: First, we define a direction template w.r.t the predicted box as shown in Fig. 3. The regions of 8 different directions are separated. The direction template is with the same size as the det-box. Then the mask patches of 8 directions are processed in parallel. For each direction, the mask patches which belong to this direction will be extracted by finding their connected components. For each mask patch, the occupancy rate (O_r) will be calculated to confirm whether this patch is in the corresponding direction to the center object. The occupancy rate is defined as follows:

$$O_r = \frac{Mask_d \cap Mask_t}{Mask_d} \tag{1}$$

where $Mask_d$ is the binary mask of the direction patch and $Mask_t$ is the mask of the corresponding direction in the direction template. If O_r is higher than a threshold O_t(0.2 in our experiments), the direction patch is regarded as part of the center object's mask. This method can tolerate det-box's error within limits. After the direction patches of 8 directions which belong to the center obje ct are collected, we can get a coarse mask of the object by piecing them together. In order to fill the small holes in the mask, we apply a close morphology operation on the coarse mask to refine the mask.

To illustrate the mechanism of how to use direction map to reject the mask patches which are not part of the center object more clearly, we supply an example. As shown in stage 2 of Fig. 3, the rightmost person's det-box contains the masks of both its center object and another person near him. Take the direction in red as an instance, two mask patches which are in the region of the direction are predicted as other directions. Therefore, it is apparent that those two mask patches are not from the center objects. The final instance mask does not contain these mask patches.

3.8 Training and Loss Functions

Given the dataset which is annotated with instance masks, the bounding box, semantic segmentation targets and direction segmentation targets can be extracted from the instance annotations. The loss function used to optimize the model is the sum of three sub-task loss functions including object detection (L_{det}), semantic segmentation (L_s), and direction segmentation (L_d). The loss is defined as follows:

$$L = L_{det} + \alpha L_s + \beta L_d \tag{2}$$

where α and β are the scale factors to reweight the three loss functions and in our experiments, the α and β are set to 100.

As for the object detection loss, we adopt the same loss proposed in SSD [23]. We adopt the same data augmentation and matching strategy suggested in the conventional SSD framework.

For both of the two segmentation tasks, the loss is the cross-entropy between the predicted and target class distribution for each pixel. To boost the segmentation performance on the hard cases in images, we adopt the bootstrapping cross-entropy proposed by Wu *et al.* [33]. Supposing C is the number of categories, and $y_1, ..., y_N \in \{1, ..., c\}$ are the target labels for the pixels $1, ..., N$, and $p_{i,j}$ is the posterior class probability for pixel i and class j, the bootstrapping cross-entropy loss can be defined as follows:

$$L_s = -\frac{1}{K}\sum_{1}^{K} log(p_k) \qquad log(p_k) \in TopK(log(p_{i,j})), y_i = j \tag{3}$$

The misclassified pixels or pixels where we predict the correct label with a small probability will be selected to be optimized in the first place depending on the number of pixels K (16384 in our experiments) that we consider.

4 Experiments

In this section, we show that our FSSI can achieve competitive performance in terms of object detection, semantic segmentation and instance segmentation with a fast speed. We implement FSSI with PyTorch[1] throughout all the experiments. We conduct experiments on PASCAL VOC [7] and MSCOCO [21] for which both bounding box annotations and segmentation maps or instance annotations are available.

[1] https://github.com/pytorch/pytorch.

4.1 Experiments on PASCAL VOC

PASCAL VOC includes 20 object categories for detection, semantic segmentation, and instance segmentation. All images in PASCAL VOC have bounding box annotations but there are only 1464 fully labeled images in the train set and another 1449 in the validation set for segmentation. We evaluate the semantic segmentation performance on the VOC2012 segmentation test set. We also augment the train set with extra annotations from the semantic boundaries dataset (SBD) [8]. Therefore, there are 10582 images in the train set (we call this set VOC12-sem-seg-aug-train). Besides, VOC07 trainval set (VOC07-trainval) has 5011 images, and VOC07 test set (VOC07-test) has 4952 images for object detection.

As for the instance segmentation task on PASCAL VOC dataset, we use the annotations in SBD [8] and follow the common protocols used in several recent works [10,17]. There are 5623 images in the train set (SBD-train) and 5732 images in the validation set (SBD-val) according to the PASCAL VOC 2012 splits. We train the models with the train set and evaluate them with the validation set as there is no annotation for the test set.

The performance of semantic segmentation and direction segmentation on PASCAL VOC dataset is mainly measured with mean intersection over union (mIoU). mAP@0.5 is used to measure the object detection quality. mAP@0.5 and mAP@0.7 are used to evaluate the performance of instance segmentation task.

While training models on the PASCAL VOC dataset, we train the FSSI on two Nvidia 1080Ti GPUs with batch size 32 for 60 epochs with initial learning rate 0.01. We decay the learning rate with cosine annealing which is proposed by [26] for each epoch.

Table 1. Ablation study results for FSSI. The performance is evaluated on the SBD-val set. **DP**: Initialize FSSI with detection model pre-trained on VOC07-trainval and VOC07-test. **BTL**: Replace the conventional cross-entropy loss with bootstrapping cross-entropy loss. **MR**: The outputs of semantic segmentation are used to generate the direction predictions. **TP**: Adopt two-path architecture to produce segmentation results. **MT**: Double the epoches in the training phase. **DET**: Object detection. **SEM**: Semantic segmentation. **DRT**: Direction segmentation. **INS**: Instance segmentation.

DP	BTL	MR	TP	MT	DET mAP@0.5	SEM mIoU	DRT mIoU	INS mAP@0.5	INS mAP@0.7
				✓	67.39	68.28	54.29	50.12	34.95
✓					74.00	69.02	55.82	53.99	37.77
✓	✓				73.50	70.60	56.63	55.44	40.79
✓	✓	✓			**74.85**	70.84	57.14	56.47	40.88
✓	✓	✓		✓	74.07	69.56	57.33	56.70	41.01
✓	✓	✓	✓		74.02	**71.30**	**58.23**	**58.40**	**41.95**

Ablation on PASCAL VOC. We run a lot of ablation studies to analyze FSSI on the structure design and training strategy. Results are shown in Table 1 and discussed as follows.

Extra Data for Detection from VOC07 Set: As illustrated in Fig. 1, our FSSI is designed for multitasks. Therefore we want to figure out whether more data of bounding box annotations is beneficial to the performance of FSSI. In Table 1, we note that training model with VOC07-trainval and VOC07-test set can improve the performance of FSSI effectively.

Bootstrapping Cross-Entropy vs. Cross-Entropy. As mentioned in [33], continuing to learn from the pixels which are easy to be classified can not improve the segmentation performance. The model should focus on the hard pixels (edge of objects or complex surfaces) during training. In order to enhance the ability of the model to discriminate the edge of the center object, we adopt the bootstrapping cross-entropy to mine the hard pixels during the training phase. We compare the effect of conventional cross-entropy and bootstrapping cross-entropy. As shown in Table 1, bootstrapping cross-entropy can improve the mIoU of both semantic segmentation and direction segmentation, which contributes to the progress of instance segmentation performance.

Direction Segmentation Takes Semantic Results as a Reference or Not: As shown in Fig. 1, the output of semantic segmentation is used as part of feature maps to generate the direction segmentation. In order to evaluate the effectiveness of this design, we remove this link and compare their performance. As shown in Table 1, approximate 1% improvement can be obtained while taking the output of semantic segmentation as the reference of the direction segmentation.

Adopt Two Path Architecture or Not for Segmentations: In order to illustrate the effectiveness of the two-path structure, we compare the one-path and two-path structure. As shown in Table 1, we can observe the improvement with a large margin while adopting the design of two-path architecture.

Effectiveness of the Three Sub-task Design. In this section, we evaluate gains from the three sub-tasks. We evaluate the instance segmentation performance in different ways as shown in Table 2. It is obvious that the direction segmentation task can improve the performance effectively. Besides, if we replace the predicted mask with the ground truth, the performance can be improved further, which means that future advances in semantic segmentation methods can further improve the performance of our approach.

Table 2. The effectiveness of the three sub-tasks for instance segmentation. **box**: The region of box is regarded as the instance mask. **sem-mask**: The semantic mask in the box is regarded as the instance mask. **direct-mask**: The direction mask joins the post-process. **gt-sem**: The semantic mask is replaced by the ground truth. **gt-direct**: The direction mask is replaced by the ground truth.

box	sem-mask	direct-mask	gt-sem	gt-direct	mAP@0.5	mAP@0.7
✓					33.51	7.93
✓	✓				53.20	36.25
✓	✓	✓			56.70	41.00
✓		✓	✓		65.77	54.11
✓	✓			✓	62.28	51.63

Table 3. Object detection and semantic segmentation evaluation on PASCAL VOC dataset.

method	*BaseModel*	*input size*	mAP_{50}
SSD300[23]	VGG16	300	77.2
RSSD300[14]	VGG16	300	78.5
DSSD300[30]	Res101	321	78.6
Blitz300[6]	Res50	300	79.1
FSSD300[18]	VGG16	300	78.8
FSSI(ours)	Res50	300	**79.6**

(a) Detection results of PASCAL VOC2007 test set. Blitz300 and FSSI are trained with mask annotations.

method	mAP_{50}
FCN[25]	62.2
DeepLab[3]	71.6
DeconvNet[27]	72.5
Blitz300[6]	72.8
GCRF[32]	73.2
DPN[24]	74.1
FSSI(ours)	**75.1**

(b) Semantic segmentation results of PASCAL VOC2012 segmentation test dataset.

Results on PASCAL VOC

Object Detection. We first evaluate our FSSI on the object detection task on PASCAL VOC 2007 test. Our FSSI is trained with VOC2007 trainval and VOC2012 trainval. Apart from the bounding box annotations, semantic segmentation labels of VOC2012 are also used to optimize the FSSI model. As shown in Table 3(a), our FSSI can achieve 79.6 mAP@0.5, and outperform the BlitzNet which is also trained with extra semantic mask annotations.

Semantic Segmentation. Since semantic segmentation is one task of our FSSI, we also evaluate the performance of FSSI with other semantic segmentation algorithms. As shown in Table 3, even though the input image of FSSI is only 384×384 pixels, FSSI still can produce competitive results with other methods (http://host.robots.ox.ac.uk:8080/anonymous/BHMCV1.html).

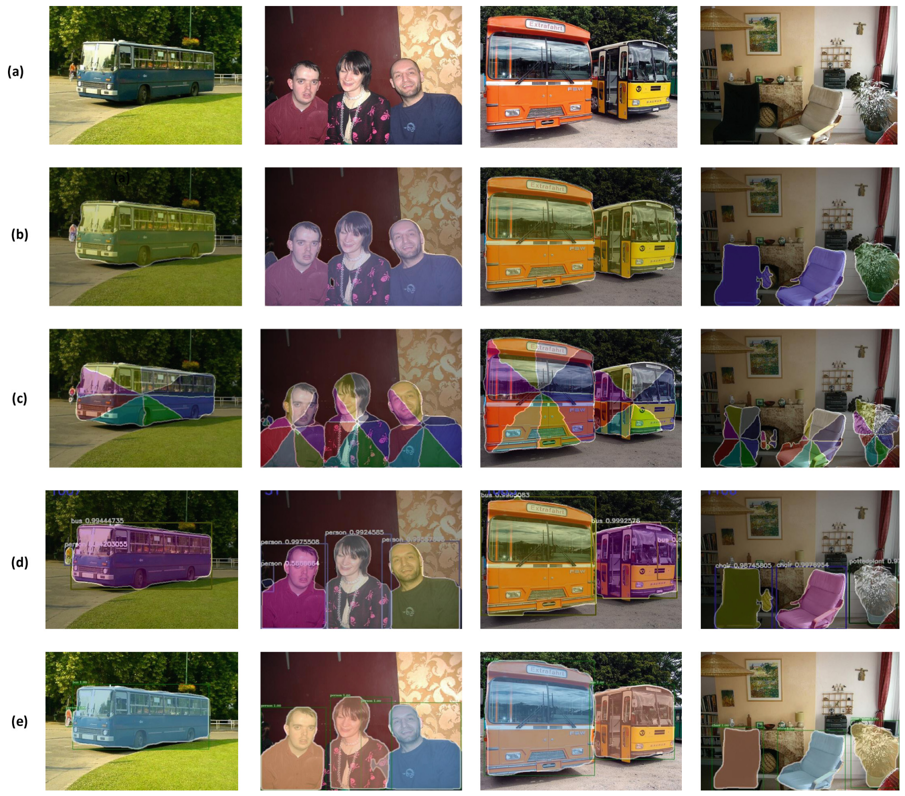

Fig. 4. Instance segmentation results. (a) Input image. (b) Semantic segmentation results. (c) Direction prediction results. (d) Instance segmentation results of our FSSI. (e) Instance segmentation results of MaskRCNN.

Instance Segmentation. We compare the results of our FSSI with the state-of-the-art approaches on PASCAL VOC 2012 dataset. As shown in Table 4, note that our approach can achieve better performance than MaskRCNN with the approximate same size of input images, even though our FSSI cannot catch up with other complex models such as MNC, FCIS, and BAIS which adopt input images with higher resolution. There are some sample results on the VOC2012 validation set as shown in Fig. 4. It is evident that our model produces competitive results with MaskRCNN[2]. Moreover, our FSSI can produce more refined masks than MaskRCNN.

[2] This result is produced by ourselves based on https://github.com/roytseng-tw/Detectron.pytorch.

Table 4. Evaluation of different methods on the PASCAL VOC2012 validation set.

Method	Input size	mAP@0.5	mAP@0.7	Time/img (ms)
SDS [10]	-	49.7	25.3	48000
PFN [19]	-	58.7	42.5	1000
InstanceFCN [4]	-	61.5	43.0	1500
MNC [5]	600˜	63.5	41.5	360
FCIS [17]	600˜	66.0	51.9	230
BAIS [11]	-	65.7	48.3	780
MaskRCNN [12]	300˜500	56.7	39.2	53
FSSI (ours)	384 × 384	58.4	42.0	17

4.2 Inference Time Comparison

As shown in Table 4, our FSSI only consumes 17 ms (network inference + post-process) for each input image averagely on the Nvidia 1080Ti GPU, which is about 4 times faster than Mask-RCNN with the approximate same size of input images.

4.3 Microsoft COCO Dataset

We further evaluate FSSI on the MSCOCO dataset. MSCOCO dataset includes 80 categories for detection and instance segmentation. There are no official semantic annotations for training and validation. In this work, we generate the semantic segmentation targets by combining instances of one category. As shown in Table 5(a), our FSSI can reach 53.8 mIoU with 384 × 384 input, which is better than BlitzNet whose input size is 512 × 512. As for the instance segmentation, our FSSI can achieve 17.5 mAP@[0.5:0.95], which is competitive with the MaskRCNN.

Table 5. Results of MSCOCO dataset. The *mmAP* denotes mAP@[0.5:0.95].

method	*Backend*	*input*	*mIoU*
Blitznet[6]	Res50	512	53.5
FSSI(ours)	Res50	384	53.8

(a) Semantic segmentation results on COCO validation set.

method	*Backend*	*input*	*val mmAP*	*test mmAP*
MaskRCNN[12]	Res50	300˜500	18.1	18.3
FSSI(ours)	Res50	384	17.5	17.8

(b) Instance segmentation results on COCO validation and test-dev set.

5 Conclusions

In this work, we propose fast single shot instance segmentation (FSSI), a lightweight and fast framework for instance segmentation. The paradigm represents the instance segmentation task by a combination of three sub-tasks: object detection, semantic segmentation and direction prediction. We have designed a new FCN architecture that utilizes this paradigm. The three sub-tasks share the fusion feature generated from the base model. Two path architecture is also adopted to supply feature maps with low-level information to produce more accurate masks. The proposed framework can run at 4 times faster than MaskRCNN while having the competitive performance on PASCAL VOC and MSCOCO.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 61471123).

References

1. Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 328–335 (2014)
2. Arnab, A., Jayasumana, S., Zheng, S., Torr, P.H.S.: Higher order conditional random fields in deep neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 524–540. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_33
3. Chen, L., 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)
4. 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
5. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: CVPR (2016)
6. Dvornik, N., Shmelkov, K., Mairal, J., Schmid, C.: BlitzNet: a real-time deep network for scene understanding. In: ICCV (2017)
7. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
8. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: 2011 International Conference on Computer Vision, pp. 991–998 (2011)
9. Hariharan, B., Arbelaez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: CVPR (2015)
10. 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
11. Hayder, Z., He, X., Salzmann, M.: Boundary-aware instance segmentation. In: CVPR (2017)
12. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
14. Jeong, J., Park, H., Kwak, N.: Enhancement of SSD by concatenating feature maps for object detection. CoRR abs/1705.09587 (2017)

15. Kong, T., Yao, A., Chen, Y., Sun, F.: HyperNet: towards accurate region proposal generation and joint object detection. In: CVPR, pp. 845–853 (2016). 00026
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
17. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: CVPR (2017)
18. Li, Z., Zhou, F.: FSSD: feature fusion single shot multibox detector. CoRR abs/1712.00960 (2017)
19. Liang, X., Wei, Y., Shen, X., Yang, J., Lin, L., Yan, S.: Proposal-free network for instance-level object segmentation. CoRR abs/1509.02636 (2015)
20. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. CoRR abs/1612.03144 (2016)
21. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48. 01470
22. Liu, S., Qi, X., Shi, J., Zhang, H., Jia, J.: Multi-scale patch aggregation (MPA) for simultaneous detection and segmentation. In: CVPR (2016)
23. 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
24. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: ICCV (2015)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
26. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with restarts. CoRR abs/1608.03983 (2016)
27. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV (2015)
28. Pinheiro, P.O., Lin, T.-Y., Collobert, R., Dollár, P.: Learning to refine object segments. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 75–91. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_5
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 91–99. Curran Associates, Inc. (2015)
30. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: DSOD: learning deeply supervised object detectors from scratch. In: ICCV (2017)
31. Uhrig, J., Cordts, M., Franke, U., Brox, T.: Pixel-level encoding and depth layering for instance-level semantic labeling. In: Rosenhahn, B., Andres, B. (eds.) GCPR 2016. LNCS, vol. 9796, pp. 14–25. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45886-1_2
32. Vemulapalli, R., Tuzel, O., Liu, M.Y., Chellapa, R.: Gaussian conditional random field network for semantic segmentation. In: CVPR (2016)
33. Wu, Z., Shen, C., van den Hengel, A.: Bridging category-level and instance-level semantic image segmentation. CoRR abs/1605.06885 (2016)
34. Zhang, Z., Fidler, S., Urtasun, R.: Instance-level segmentation for autonomous driving with deep densely connected MRFs. In: CVPR (2016)
35. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)