Pixel Offset Regression (POR) for Single-shot Instance Segmentation

Yuezun Li[1], Xiao Bian[2], Ming-ching Chang[1], Longyin Wen[2] and Siwei Lyu[1]
[1]University at Albany, State University of New York, NY, USA
[2]GE Global Research Center, Niskayuna, NY, USA

Abstract

State-of-the-art instance segmentation methods including Mask-RCNN and MNC are multi-shot, as multiple region of interest (ROI) forward passes are required to distinguish candidate regions. Multi-shot architectures usually achieve good performance on public benchmarks. However, hundreds of ROI forward passes in sequel limits their running efficiency, which is a critical point in several utilities such as vehicle surveillance. As such, we arrange our focus on seeking a well trade-off between performance and efficiency. In this paper, we introduce a novel Pixel Offset Regression (POR) scheme which can simply extend single-shot object detector to single-shot instance segmentation system, i.e.*, segmenting all instances in a single pass. Our framework is based on VGG16*[1] *with following four parts: (1) a single-shot detection branch to generate object detections, (2) a segmentation branch to estimate foreground masks, (3) a pixel offset regression branch to effectively estimate the distance and orientation from each pixel to the respective object center and (4) a merging process combining output of each branch to obtain instances. Our framework is evaluated on Berkeley-BDD, KITTI and PASCAL VOC2012 validation set, with comparison against several VGG16 based multi-shot methods. Without whistles and bells, our framework exhibits decent performance, which shows good potential for fast speed required applications.*

1. Introduction

Instance segmentation can be viewed as a joint task of object detection and semantic segmentation. Given an input image, it assigns a class (*semantic*) label as well as an instance (*identity*) label for each pixel. Instance segmentation has recently achieved significant progress based on the *convolutional neural network* (CNN). However, the problem remains challenging due to the large variability in object categories, the amount of labeled data, and the preferred running time. State-of-the-art methods including the Mask-RCNN [14] and the Multi-task Network Cascades (MNC) [6] generate candidate object proposals using the Region Proposal Network (RPN) [21], and perform instance segmentation in a "multi-shot" fashion. Shared features from the given image are first extracted in a base network, and then each region proposal is passed to a sub-network for region refinement. The drawback of this "multi-shot" architecture is that each region proposal must go through the sub-network separately before the final segmentation can be obtained. Their running efficiency can be reduced, since typically hundreds to thousands of region proposals are required [21]. Notice that running efficiency of instance segmentation is a critical point in several utilities such as vehicle surveillance. Therefore, pursuing good running efficiency is highly deserved and necessary. To this end, we propose a novel Pixel Offset Regression (POR) scheme, which can simply extend single-shot detector to single-shot instance segmentation system, *i.e.*, segmenting all objects using a single forward pass. Our method exhibits well trade-off between performance and efficiency.

The recent Single-Shot Multibox Detector (SSD) [17] exhibits great performance for real-time object detection using a fully convolutional network. SSD eliminates the need of object region proposal generation. Instead, a set of *anchor boxes* over different scales and aspect ratios are used in place of the region proposals. SSD calculates a matching score for each anchor box regarding a candidate object, and adjusts the box to match the object shape. The detection of multiple objects of multiple classes can be completed within a single network pass.

Our framework is based on VGG16 with following four major parts as shown in Fig.1: (1) a single-shot *detection branch* to generate object detection proposals, (2) a *segmentation branch* to estimate object foreground masks, (3) a pixel offset regression branch to achieve per-pixel object center regression, and (4) a *merging process* combining the output of each branch to infer the final instances. In our single-shot pipeline, the base network extracts features that are fed into a detection branch similar to the SSD workflow. The detection branch generates object localization as

[1]This paper aims to shed light on the generality of POR scheme for instance segmentation. We leave advanced models integration (ResNet-50, -101 and -152) for top performance to the future work.

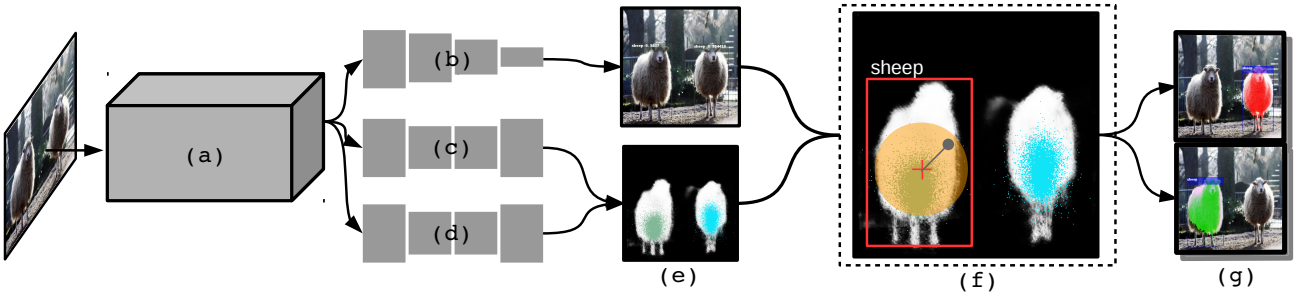

Figure 1. Overview of our framework. (a) is the base-net using VGG16 without fc6 and fc7. (b)(c)(d) are detection branch, segmentation branch and pixel offset regression branch respectively. (e) are predicted detection, foreground mask and regressed object center location of pixels. (f) is the merging process. Pixels with regressed object center in orange area will be taken as instance. (g) are instance segmentation results.

bounding boxes together with the class and instance labels. Combining with foreground segmentation, a mask inside bounding box can be cropped out. However, different objects in this mask can not be separated. To further segment out object instances in a single-shot, we add a pixel offset regression branch from the base network to distinguish multiple instances. Pixel offset regression scheme estimates a per-pixel offset vector pointing from each pixel (within the foreground mask) to the object center [2]. Finally, we use a merging process to combine the results from each branches to generate the instance segmentation.

We highlight three contributions of our work:

- Inspired by bounding box regression, We propose a dense regression scheme: pixel offset regression, which adjust each foreground pixel pointing to corresponding object center.
- With the aid of pixel offset regression scheme, we can simply extend single-shot object detector to single-shot instance segmentation system.
- The single-shot nature of our framework makes it fast (∼8 FPS without code optimization on NVIDIA K40 GPU), which shows a good potential for high efficiency required applications.

2. Related Work

Object Detection. The task is to predict a bounding box around each object of interest with a class label. R-CNN based methods [10, 9, 21] are the most predominant deep networks for object detection in recent years. The "vanilla" R-CNN [10] follows an object proposal generation paradigm: First, a large amount of region proposals are generated using *selective search* [25], followed by a fixed-size wrapping of each region of interest (ROI). Next each warped ROI is respectively passed into a CNN to obtain a classification score. To eliminate the bottleneck of object proposal generation, Faster-RCNN [21] uses a RPN to substitute selective search, where the feature maps are shared across all object proposals. The ROI pooling layer warps each ROI for sub-network feeding. R-CNN and Faster-RCNN are multi-shot architectures. In comparison, the Single-Shot Multibox Detector (SSD) [17] performs detection in a single network pass, which significantly increases the speed. However, SSD yields detection boxes that can only localize objects. Thus it is not suitable for applications need fine-grain mask in scene parsing and segmentation.

Semantic Segmentation. The task is to assign a class label for each pixel of the input image. Recent methods [27, 4, 18, 20, 3] are all based on the *fully convolutional network* (FCN) [19], where the fully connected layers in the CNN are replaced with fully convolutional layers. As higher layers in the FCN contain more semantic information but less spatial cues due to max-pooling, FCN adopts a hierarchical training scheme that fuses the 32-stride, 16-stride, and 8-stride layers in turns. Noh *et al*. [20] propose a coarse-to-fine deconvolution network to enhance the semantic features while retaining spatial contexts. Liu *et al*. [18] combine global average pooling within the layers in FCN to observe global contexts. To compensate the loss of spatial cues in the pooling, The work of DeepLab [3] introduces the *dilated convolution* to increase the receptive field while maintaining the output resolution. Their scale-awareness work [4] takes advantage of image-level scale priors with the assistance of an attention model. Hengshuang *et al.* [27] propose a pooling pyramid on the layers to embed various scale semantic features in the FCN to differentiate local and global contexts. Despite these methods perform well on pixelwise semantic segmentation, they cannot distinguish individual instances of objects.

Instance Segmentation. Instance segmentation is a joint task combining detection and segmentation, with an aim to simultaneously assign both the class and instance labels for object foreground pixels. Existing methods [14, 12, 13, 6] typically follow the R-CNN object detection pipeline,

[2] Our object center regression is analogical to the way SSD matches anchor boxes to candidate objects, where the box centers and sizes are estimated. In our method, the per-pixel offset vectors *w.r.t.* the object center are estimated.

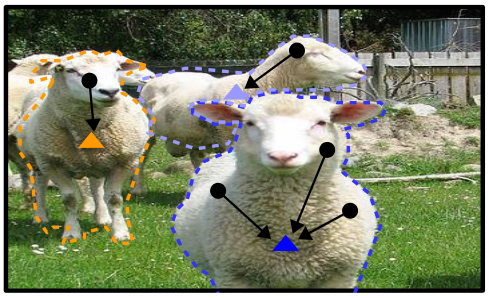

Figure 2. Illustration of pixel offset regression. The triangles are center location of respective objects. The black arrow denotes adjust each pixel inside object pointing to corresponding object center.

which is multi-shot in nature. Dai *et al*. [6] proposed a multi-task network cascade (MNC) model that delineates object masks, classifies objects, and distinguishes object instances in three separated steps. The Mask-RCNN of Kaiming *et al*. [14] adds a small FCN branch to the Faster-RCNN to predict object masks. A ROI-align pooling keeps track of pixel locations during the pooling to alleviate the loss of spatial details. Another line of works adds a conditional random field [2] or a metric similarity model [8] to the end of the instance segmentation network to refine their results. Such post-processing approaches are time-consuming and not capable in handling complex scenes. The PFN [16] performs instance segmentation by clustering without object detection. FCIS [15] identifies instances using "position-sensitive score maps", where each channel of the map represents different positions of a ROI in a 3×3 grid, and obtains object classification at the end. FCIS uses ResNet-101 as the base network and many GPU resources are used in training. In this paper, we focus on highlighting the generality of Pixel Offset Regression (POR) scheme, which can extend single-shot object detector to single-shot instance segmentation system. We leave advanced model integration for top performance to the future work.

3. Method

Our single-shot instance segmentation pipeline consists of five components as shown in Fig.1: a *base network* followed by a *detection branch*, a *segmentation branch* and a *pixel offset regression branch*, where the outputs are combined in a *merging process*. Boosting top performance with advanced models such as ResNet-101 is not our key in this paper, as we focus on the generality of proposed Pixel Offset Regression (POR) scheme. Considering training cost and stability, We adopt VGG16 network [24] pre-trained on the ILSVRC CLS-LOC dataset [23] without the last two fully connected layers `fc6`, `fc7` to be our base network. We describe the rest of the components in the following subsections.

The performance of our method depends on how well the object segmentation and the class/instance labels are. To achieve the best single-shot performance after the feature extraction from the base network, we use a fast detection branch to determine the object bounding boxes with class / instance labels, in parallel with segmentation branch and a pixel offset branch.

Our detection branch follows the single-shot multi-class object detection pipeline (*e.g.* SSD [17]) that effectively generates object bounding boxes with class labels. Our method can be reduced to SSD by keeping the base and detection branch (with the segmentation branch, pixel offset regression branch and merging process off).

Our segmentation branch determines foreground regions. With the aid of detections, a mask inside detection bounding box can be cropped out. However, it is challenging to distinguish different instance in this mask, thus per-pixel instance labeling is non-trivial. The determination of instance labels of segmented regions is the key problem in instance segmentation. Our solution is to use a *deconvoluted* fully convolutional network (FCN) in our pixel offset regression branch, inspired from the work of U-Net [22]. The architecture consists of concatenated convolutional and deconvolutional layers. The advantage of such conv-deconv FCN is that it can effectively explore the spatial and semantic information within the image. This branch performs the object center regression as shown in Fig.2.

§3.1 will describe our pixel offset regression, which assigns the class and instance labels for each foreground pixel. This step is complementary to and works hand-in-hand with our detection and segmentation branch to effectively determine object instances. As Fig.1(f) depicts, the per-pixel regression vectors point to respective object center(s) in dense clusters. §3.2 will describe how the merging of the detection, segmentation and pixel offset regression branches in determining the final instance segmentation.

3.1. Pixel Offset Regression

Our per-pixel object center regression estimates a vector pointing from each pixel to its respective object center. Let $P = \{p_i \mid i = 1, 2, ..., n\}$ denotes all pixels in the input image, and $O = \{o_j \mid j = 1, 2, ..., m\}$ denotes the set of objects (instances) appeared in the given image. Let $\mathbf{v}_i$ denotes the location of pixel p_i, and $\mathbf{c}_j$ denotes the center location of object o_j. We enforce each pixel to belong to exactly one object instance or the background. If pixel p_i is inside object o_j, $p_i \in o_j$; otherwise p_i is a background pixel, $p_i \in \emptyset$, the case of an outlier for the regression. The offset $\mathbf{f}_i$ of pixel p_i pointing to object center $\mathbf{c}_j$ is then:

$$\mathbf{f}_i = \begin{cases} \mathbf{c}_j - \mathbf{v}_i, & p_i \in o_j, \\ 0, & p_i \in \emptyset, \end{cases} \tag{1}$$

Fig.1(f) shows the accumulation of all center-pointing vectors $\{\mathbf{f}_i\}$ form dense clusters, where each indicates an object instance. The next section describes how we determine

the object instances using a voting scheme in the merging process.

We design an adaptive voting zone for each detection center using a threshold scheme, as in Fig.1(f). Let $\mathbf{k}_j$ denotes the diagonal length of detection box d_j. The adaptive voting zone is a circle with radius $r_j = \sigma \cdot \mathbf{k}_j$. Let $\phi_{ij} = \{0, 1\}$ be an indicator denoting whether pixel p_i belongs to object o_j or not. Let P_f denote the set of pixels from the object foreground mask. The thresholding scheme in selecting object instance pixels is formulated by defining ϕ_{ij} as:

$$\phi_{ij} = \begin{cases} 1, & \hat{\mathbf{c}}_i - \mathbf{b}_j <= r_j \text{ and } p_i \in P_f, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Our merging process ensures that each pixel in the input image (including object pixels and the background) is assigned with the best estimated class and instance labels, such that a sound instance segmentation is obtained.

3.2. Merging Process for Instance Estimation

The merging process combines results from each branch to produce the instance segmentation. Ideally, object centers indicated by all $\{\mathbf{f}_i\}$ should be dense and sharp, provided that: (1) each object is well localized from the detection network, (2) the foreground mask is well obtained from the segmentation network, and (3) the regression has a good fit. In practice, the regressed center-pointing vectors $\mathbf{f}_i$ form a distribution around the center. We use a simple effective thresholding scheme to robustly estimate the true object center as well as determine the pixels belonging to each object instance. We assume each object o_j can be successfully detected in a bounding box d_j, where $\mathcal{D} = \{d_j \mid j = 1, 2, ..., m\}$ is the set of object detections. For each center-pointing vector $\mathbf{f}_i$, we recover its object center $\hat{\mathbf{c}}_i$ using Eq.(1). Let $\mathbf{b}_j$ denote the center of box d_j. We calculate the Euclidean distance between $\hat{\mathbf{c}}_i$ and $\mathbf{b}_j$ to estimate the probability if pixel p_i belonging to object instance o_j.

3.3. Loss Functions

The loss L for training our network is the sum of the detection loss L_{det}, the segmentation loss L_{seg} and pixel offset regression loss L_o. The detection loss L_{det} is the weighted sum of the localization loss L_{loc} and class confidence loss L_{cf} from all matched anchor boxes: [3]

$$L_{det} = \frac{1}{N}(L_{cf} + L_{loc}), \quad (3)$$

where N is the number of matched anchor boxes. We calculate L_{seg} using pixel-wise summation of cross entropy terms. The predicted confidence map after softmax is a 2-channel feature map, denoted as $\mathcal{M}$. $\mathcal{M}_i^l$ is as the likelihood of the i-th pixel matching its ground truth class label l (0 or 1), and $|\mathcal{M}|$ is the number of pixels in $\mathcal{M}$. The class loss L_{seg} is defined as:

$$L_{seg} = \frac{1}{|\mathcal{M}|}\left(-\sum_i \log(\mathcal{M}_i^l)\right). \quad (4)$$

We use an adaptive weighted sum of smooth L1 loss [9] to define our pixel offset regression loss L_o. We denote the center-pointing pixel offset map as $\mathcal{N}$ and the corresponding ground truth map as $\mathcal{N}^g$. The loss L_o is defined as:

$$L_o = \frac{1}{|P_f|}\left(\sum_i \gamma_i \cdot \text{smooth}_{L_1}(\mathcal{N}_i, \mathcal{N}_i^g)\right), \quad (5)$$

where the weight $\gamma_i >= 0$ controls the contribution of loss from each foreground pixel, and $|P_f|$ is the number of all foreground pixels.

Existing methods including SSD [17] and Faster R-CNN [21] calculate the smooth L1 loss without considering object scales. In comparison, we propose an improvement by calculating an **adaptive weighting** of our pixel-wise offset loss L_o, with an aim to balance the scale variations from large vs. small objects in the scene during the object center regression. This design is important as it can prevent the domination of large objects (and ignoring the small objects) that occurs in instance regression,which is due to two reasons: (1) the greater amount of pixels in large objects can dominate the regression, and (2) large objects contain larger pixel-to-object-center offsets especially at the boundary pixels. Our adaptive weighting scheme can effectively balance loss contributions from all objects neglecting their sizes. Let w_j and h_j be the width and height of the bounding box of an object o_j. The adaptive weight γ_i is defined by:

$$\gamma_i = \begin{cases} (\frac{1}{w_j\, h_j})^{3/2}, & \text{if } p_i \in o_j, \\ 0, & \text{if } p_i \in \emptyset. \end{cases} \quad (6)$$

3.4. Implementation Details

Our detection network is based on the setting of SSD300 [17]. As shown in Fig.1, we use 6 conv layers after the base network and 6 branching paths from these layers to the end to generate multi-scale object detections. Our segmentation branch and pixel offset regression branch shares most layers of a deconvoluted FCN based on the U-Net [22], where the original down sampling path of U-Net is replaced by our base network. We upsample in each deconv layer to match the resolution of the corresponding conv layer. Outputs from the deconv layer and the corresponding conv layer are concatenated as input to the next layer during an up-sampling. Such up-sampling repeats until the resolution of last layer matches the input image (300×300). Then two paths at the last layer generate the semantic confidence map $\mathcal{M}$ ($300 \times 300 \times 2$) and the pixel offset map $\mathcal{N}$

[3] We make our network loss consistent with SSD [17], so our method can be reduced to a SSD detector.

Table 1. Instance segmentation performance (using the mAP metric at 0.5 and 0.7) and running speed (FPS) of different VGG16 based methods on pascal VOC2012 validation set.

Type	Methods	0.5	0.7	GPU	Platform	FPS
-	PFN [16]	58.7	42.5	Titan	C++	1.0
M	MNC [6]	63.5	41.5	K40	C++	2.4
	SDS [12]	43.8	21.3	-	C++	<1.0
	Chen [5]	46.3	27.0	-	C++	<1.0
S	Ours-300	49.1	27.8	K40	Python	**7.7**
	Ours-512	54.7	28.1	K40	Python	4.0

($300 \times 300 \times 2$). We set voting scale parameter $\sigma = 0.17$ in merging process.

Training. We conduct experiments on a workstation equipped with dual Intel Xeon X5570 2.93 GHz 4 Core CPU and a NVIDIA Tesla K40 GPU. We use *adadelta* optimization with training batch size 8. Learning rate is initially 0.01, decreased by a factor of 10 after every 20 epochs. To improve training stability, we start with the training of the entire network for 100 epochs (considering the overall losses $L = L_{det} + L_{seg} + L_o$), and then alternatively train individual branches. To ensure accurate object detection for the training of the segmentation network, we first fine-tune the detection branch with learning rate starting from 0.001 for 50 epochs. We then fine-tune the segmentation branch with fixed detection parameters for 50 epochs. These two steps iterate until the epoch reaches 200.

4. Experiments

In this section, we describe the evaluation experiments in detail on three datasets: PASCAL VOC 2012, KITTI and Berkeley-BDD. Fig.3 illustrates a few of our instance segmentation examples.

4.1. PASCAL VOC 2012

The PASCAL VOC2012 instance segmentation benchmark [7] consists of 20 object categories, 1464 training images, 1449 validation images, and 1456 testing images. Following the [6, 15] protocol, we use the VOC2012 training set with additional instance annotations provided by [11] as our training set. Experiments are performed on the VOC2012 validation set using the mean average precision (mAP) [12] as the evaluation metric. We use the intersection-over-union (IoU) thresholds at 0.5 and 0.7 to compare the resulting segmentation mask against the groundtruth for each object.

Main Results. Table 1 compares our method with recent VGG16 based instance segmentation methods on the PASCAL VOC2012 validation set at $\text{mAP}_{0.5}$ and $\text{mAP}_{0.7}$. The "-" in PFN [16] architecture denotes it is neither multi-shot nor single-shot, as it uses clustering algorithm as post-process to generate instance. Ours-300 is our framework with input size 300×300, which outperforms the multi-shot methods SDS [12] and Chen *et al.* [5] at $\text{mAP}_{0.5}$ by 4.9% and 2.4%, respectively. It also outperforms them at $\text{mAP}_{0.7}$ by 6.4% and 0.7%. Despite the PFN [16] and multi-shot method MNC [6] can achieve better performance, the running time efficiency limits their potential in many applications such as vehicle surveillance.

Table 2. Instance segmentation performance at mAP 0.5 of *car*, *person* and *bus* on KITTI and Berkeley-BBD.

Category	KITTI	Berkeley-BBD
car	50.0	41.7
person	13.4	9.6
bus	37.6	23.5

Ablation study. Ours-512 in Table 1 is the framework with input size 512×512. We detached the shared layers in each branch of this framework into three independent branches. We modify segmentation branch to predict semantic mask of image. The results show the performance at $\text{mAP}_{0.5}$ and $\text{mAP}_{0.7}$ are increased by $5.6\%, 0.3\%$ compared to Ours-300, yet FPS is affected by input size increasing and shared layers reducing.

Run time efficiency. Since SDS [12] and Chen *et al.* [5] utilizes time consuming bottom-up proposal generation, we do not involve them in running efficiency comparison (denote as < 1.0). Despite we do not highly optimize our code, our method shows well trade-off between performance and running efficiency, thus shows the potential for fast speed needed applications.

4.2. KITTI and Berkeley-BDD

KITTI instance-level dataset [1] consists of 200 semantically annotated train as well as 200 test images with 34 categories. Berkeley-BDD dataset [26] for segmentation contains 3333 images in training set, 745 images in validation set and 1483 images in testing set, with 40 categories.

We evaluate our framework on three categories *car*, *person* and *bus* on training set of KITTI and validation set of Berkeley-BDD using pre-trained model on PASCAL VOC 2012 at mAP 0.5, which is shown in Table 2.

5. Conclusions

We present a novel Pixel Offset Regression (POR) scheme to extend single-shot object detector to single-shot instance segmentation system. The POR scheme can effectively cluster foreground pixels *w.r.t.* their object centers and solve the instance segmentation problem using a single forward pass. Our framework is based on VGG16 and compared favorably with VGG16 based multi-shot methods, and exhibits potential for use in high efficiency applications such as vehicle surveillance. Future work includes

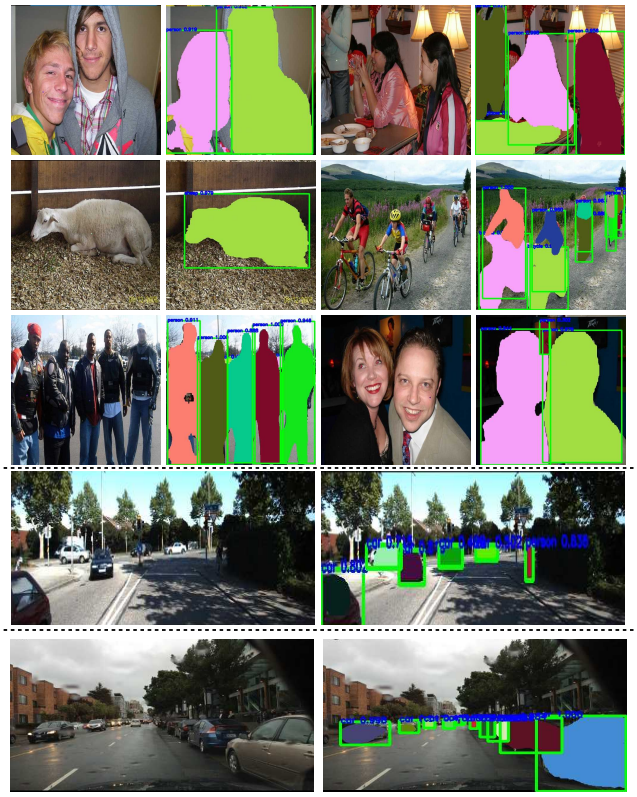

Figure 3. Visual illustration of performing our framework on PASCAL VOC2012 (first three rows), KITTI (4th row) and Berkeley-BDD (last row).

the use of more powerful models (*e.g.* ResNet) and integration of *feature pyramid network* into our segmentation branch for improved performance.

Acknowledgement. This work is supported by the United States Air Force Research Laboratory (AFRL) under Contract No. FA8750-16-C-0166.

References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *BMVC*, 2017.

[2] A. Arnab and P. H. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, 2017.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[4] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[5] Y.-T. Chen, X. Liu, and M.-H. Yang. Multi-instance object segmentation with occlusion handling. In *CVPR*, 2015.

[6] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Vlasses (VOC) challenge. *IJCV*, 2010.

[8] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv:1703.10277*, 2017.

[9] R. Girshick. Fast R-CNN. In *ICCV*, 2015.

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

[11] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

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

[13] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

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

[16] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *TPAMI*, 2017.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[18] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. In *ICLR*, 2016.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[20] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *TPAMI*, 2017.

[22] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[23] 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.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[25] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[26] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *CVPR*, pages 3530–3538, 2017.

[27] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.