# Path Aggregation Network for Instance Segmentation

Shu Liu[†] Lu Qi[†] Haifang Qin[§] Jianping Shi[‡] Jiaya Jia[†,♭]

[†]The Chinese University of Hong Kong [§]Peking University
[‡]SenseTime Research [♭]YouTu Lab, Tencent

{sliu, luqi, leojia}@cse.cuhk.edu.hk qhfpku@pku.edu.cn shijianping@sensetime.com

## Abstract

*The way that information propagates in neural networks is of great importance. In this paper, we propose Path Aggregation Network (PANet) aiming at boosting information flow in proposal-based instance segmentation framework. Specifically, we enhance the entire feature hierarchy with accurate localization signals in lower layers by bottom-up path augmentation, which shortens the information path between lower layers and topmost feature. We present adaptive feature pooling, which links feature grid and all feature levels to make useful information in each level propagate directly to following proposal subnetworks. A complementary branch capturing different views for each proposal is created to further improve mask prediction.*

*These improvements are simple to implement, with subtle extra computational overhead. Yet they are useful and make our PANet reach the $1^{st}$ place in the COCO 2017 Challenge Instance Segmentation task and the $2^{nd}$ place in Object Detection task without large-batch training. PANet is also state-of-the-art on MVD and Cityscapes.*

## 1. Introduction

Instance segmentation is one of the most important and challenging tasks. It aims to predict class labels and pixel-wise instance masks to localize a varying number of instances presented in each image. This task widely benefits autonomous vehicles, robotics, video surveillance, to name a few.

With the help of deep convolutional neural networks, several frameworks for instance segmentation, *e.g.*, [21, 33, 3, 38], were proposed where performance grows rapidly [12]. Mask R-CNN [21] is a simple and effective system for instance segmentation. Based on Fast/Faster R-CNN [16, 51], a fully convolutional network (FCN) is used for mask prediction, along with box regression and classification. To achieve high performance, feature pyramid network (FPN) [35] is utilized to extract in-network feature hierarchy, where a top-down path with lateral connections is augmented to propagate semantically strong features.

Several newly released datasets [37, 7, 45] facilitate design of new algorithms. COCO [37] consists of 200k images. Several instances with complex spatial layout are captured in each image. Differently, Cityscapes [7] and MVD [45] provide street scenes with a large number of traffic participants in each image. Blur, heavy occlusion and extremely small instances appear in these datasets.

There have been several principles proposed for designing networks in image classification that are also effective for object recognition. For example, shortening information path and easing information propagation by clean residual connection [23, 24] and dense connection [26] are useful. Increasing the flexibility and diversity of information paths by creating parallel paths following the *split-transform-merge* strategy [61, 6] is also beneficial.

**Our Findings** Our research indicates that information propagation in state-of-the-art Mask R-CNN can be further improved. Specifically, features in low levels are helpful for large instance identification. But there is a long path from low-level structure to topmost features, increasing difficulty to access accurate localization information. Further, each proposal is predicted based on feature grids pooled from one feature level, which is assigned heuristically. This process can be updated since information discarded in other levels may be helpful for final prediction. Finally, mask prediction is made on a single view, losing the chance to gather more diverse information.

**Our Contributions** Inspired by these principles and observations, we propose PANet, illustrated in Figure 1, for instance segmentation.

First, to shorten information path and enhance feature pyramid with accurate localization signals existing in low-levels, bottom-up path augmentation is created. In fact, features in low-layers were utilized in the systems of [44, 42, 13, 46, 35, 5, 31, 14]. But propagating low-level features to enhance entire feature hierarchy for instance recognition was not explored.

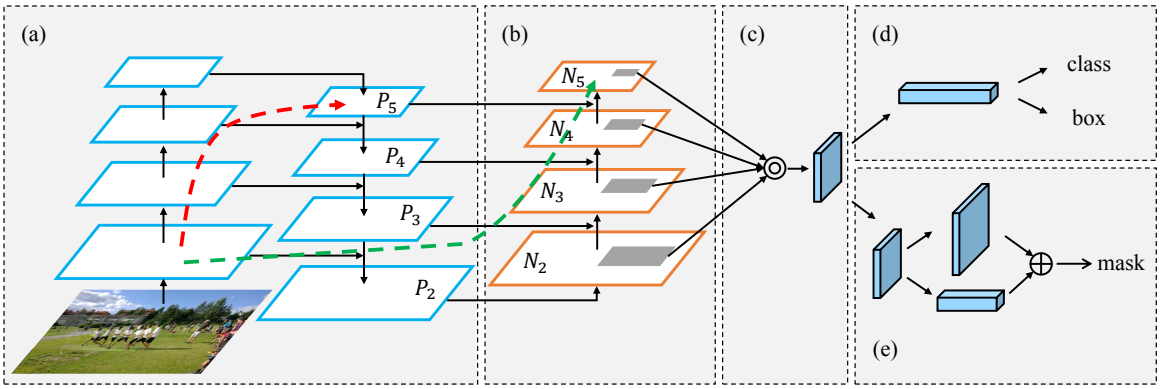


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Second, to recover broken information path between each proposal and all feature levels, we develop adaptive feature pooling. It is a simple component to aggregate features from all feature levels for each proposal, avoiding arbitrarily assigned results. With this operation, cleaner paths are created compared with those of [4, 62].

Finally, to capture different views of each proposal, we augment mask prediction with tiny fully-connected (*fc*) layers, which possess complementary properties to FCN originally used by Mask R-CNN. By fusing predictions from these two views, information diversity increases and masks with better quality are produced.

The first two components are shared by both object detection and instance segmentation, leading to much enhanced performance of both tasks.

**Experimental Results** With PANet, we achieve state-of-the-art performance on several datasets. With ResNet-50 [23] as the initial network, our PANet tested with a single scale already outperforms champion of COCO 2016 Challenge in both object detection [27] and instance segmentation [33] tasks. Note that these previous results are achieved by larger models [23, 58] along with multi-scale and horizontal flip testing.

We achieve the $1^{st}$ place in COCO 2017 Challenge Instance Segmentation task and the $2^{nd}$ place in Object Detection task without large-batch training. We also benchmark our system on Cityscapes and MVD, which similarly yields top-ranking results, manifesting that our PANet is a very practical and top-performing framework. Our code and models will be made publicly available.

## 2. Related Work

**Instance Segmentation** There are mainly two streams of methods in instance segmentation. The most popular one is proposal-based. Methods in this stream have a strong connection to object detection. In R-CNN [17], object proposals from [60, 68] were fed into the network to extract features for classification. Fast and faster R-CNN [16, 51] and SPPNet [22] sped up the process by pooling features from global feature maps. Earlier work [18, 19] took mask proposals from MCG [1] as input to extract features while CFM [9], MNC [10] and Hayder *et al*. [20] merged feature pooling to network for faster speed. Newer design was to generate instance masks in networks as proposal [48, 49, 8] or final result [10, 34, 41]. Mask R-CNN [21] is an effective framework in this stream. Our work is built on Mask R-CNN and improves it in important aspects.

Methods in the other stream are mainly segmentation-based. They learned specially designed transformation [3, 33, 38, 59] or instance boundaries [30]. Then instance masks were decoded from predicted transformation. Instance segmentation by other pipelines also exists. DIN [2] fused predictions from object detection and semantic segmentation systems. A graphical model was used in [66, 65] to infer the order of instances. RNN was utilized in [53, 50] to propose one instance in each time step.

**Multi-level Features** Features from different layers were used in image recognition. SharpMask [49], Peng *et al*. [47] and LRR [14] fused feature maps for segmentation with finer details. FCN [44], U-Net [54] and Noh *et al*. [46] fused information from lower layers through skip-connections. Both TDM [56] and FPN [35] augmented a top-down path with lateral connections for object detection. Different from TDM, which took the fused feature map with the highest resolution to pool features, SSD [42], DSSD [13], MS-CNN [5] and FPN [35] assigned proposals to appropriate feature levels for inference. We take FPN as a baseline and much enhance it.

ION [4], Zagoruyko *et al*. [62], Hypernet [31] and Hypercolumn [19] concatenated feature grids from different layers for better prediction. A sequence of operations, *i.e*., normalization, concatenation and dimension reduction, are needed to get feasible new features. In comparison, our design is much simpler.

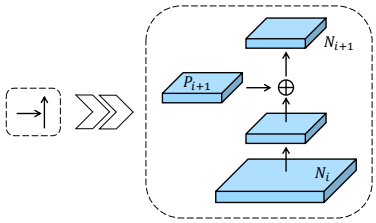


Figure 2. Illustration of our building block of bottom-up path augmentation.

Fusing feature grids from different sources for each proposal was also utilized in [52]. But this method extracted feature maps on input with different scales and then conducted feature fusion (with the max operation) to improve feature selection from the input image pyramid. In contrast, our method aims at utilizing information from all feature levels in the in-network feature hierarchy with single-scale input. End-to-end training is enabled.

**Larger Context Region** Methods of [15, 64, 62] pooled features for each proposal with a foveal structure to exploit context information from regions with different resolutions. Features pooled from a larger region provide surrounding context. Global pooling was used in PSPNet [67] and ParseNet [43] to greatly improve quality of semantic segmentation. Similar trend was observed by Peng *et al.* [47] where global convolutions were utilized. Our mask prediction branch also supports accessing global information. But the technique is completely different.

## 3. Our Framework

Our framework is illustrated in Figure 1. Path augmentation and aggregation are conducted for improving performance. A bottom-up path is augmented to make low-layer information easier to propagate. We design adaptive feature pooling to allow each proposal to access information from all levels for prediction. A complementary path is added to the mask-prediction branch. This new structure leads to decent performance. Similar to FPN, the improvement is independent of the CNN structures, such as those of [57, 32, 23].

### 3.1. Bottom-up Path Augmentation

**Motivation** The insightful point [63] that neurons in high layers strongly respond to entire objects while others are more likely to be activated by local texture and patterns manifests the necessity of augmenting a top-down path to propagate semantically strong features and enhance all features with reasonable classification capability in FPN.

Our framework further enhances the localization capability of the entire feature hierarchy by propagating strong responses of low-level patterns based on the fact that high response to edges or instance parts is a strong indicator to accurately localize instances. To this end, we build a path with clean lateral connections from the low level to top ones. This process yields a "shortcut" (dashed green line in Figure 1), which consists of less than 10 layers, across these levels. In comparison, the CNN trunk in FPN gives a long path (dashed red line in Figure 1) passing through 100+ layers from low layers to the topmost one.

**Augmented Bottom-up Structure** Our framework first accomplishes *bottom-up path augmentation*. We follow FPN to define that layers producing feature maps with the same spatial sizes are in the same network *stage*. Each feature level corresponds to one stage. We also take ResNet [23] as the basic structure and use $\{P_2, P_3, P_4, P_5\}$ to denote feature levels generated by FPN. Our augmented path starts from the lowest level $P_2$ and gradually approaches $P_5$ as shown in Figure 1(b). From $P_2$ to $P_5$, the spatial size is gradually down-sampled with factor 2. We use $\{N_2, N_3, N_4, N_5\}$ to denote newly generated feature maps corresponding to $\{P_2, P_3, P_4, P_5\}$. Note that $N_2$ is simply $P_2$, without any processing.

As shown in Figure 2, each building block takes a higher resolution feature map $N_i$ and a coarser map $P_{i+1}$ through lateral connection and generates the new feature map $N_{i+1}$. Each feature map $N_i$ first goes through a $3 \times 3$ convolutional layer with stride 2 to reduce the spatial size. Then each element of feature map $P_{i+1}$ and the down-sampled map are added through lateral connection. The fused feature map is then processed by another $3 \times 3$ convolutional layer to generate $N_{i+1}$ for following sub-networks. This is an iterative process and terminates after approaching $P_5$. In these building blocks, we consistently use channel 256 of feature maps. All convolutional layers are followed by a ReLU [32]. The feature grid for each proposal is then pooled from new feature maps, *i.e.*, $\{N_2, N_3, N_4, N_5\}$.

### 3.2. Adaptive Feature Pooling

**Motivation** In FPN [35], proposals are assigned to different feature levels according to the size of proposals. It makes small proposals assigned to low feature levels and large proposals to higher ones. Albeit simple and effective, it could generate non-optimal results. For example, two proposals with 10-pixel difference can be assigned to different levels. In fact, these two proposals are rather similar.

Further, importance of features may not be strongly correlated to the levels they belong to. High-level features are generated with large receptive fields and capture richer context information. Allowing small proposals to access these features better exploits useful context information for prediction. Similarly, low-level features are with many fine details and high localization accuracy. Making large proposals access them is obviously beneficial. With these thoughts, we propose pooling features from all levels for each proposal and fusing them for following prediction. We call this process *adaptive feature pooling*.

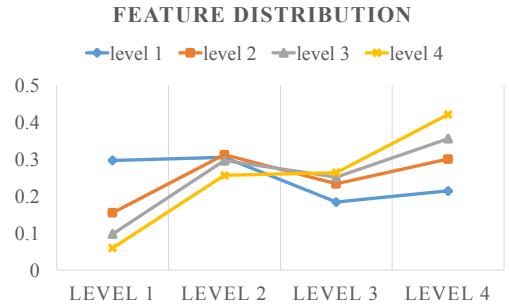


Figure 3. Ratio of features pooled from different feature levels with adaptive feature pooling. Each line represents a set of proposals that should be assigned to the same feature level in FPN, *i.e.*, proposals with similar scales. The horizontal axis denotes the source of pooled features. It shows that proposals with different sizes all exploit features from multiple levels.

We now analyze the ratio of features pooled from different levels with adaptive feature pooling. We use *max* operation to fuse features from different levels, which lets network select element-wise useful information. We cluster proposals into four classes based on the levels they were assigned to originally in FPN. For each set of proposals, we calculate the ratio of features selected from different levels. In notation, levels $1-4$ represent low-to-high levels.

As shown in Figure 3, the blue line represents small proposals that were assigned to level 1 originally in FPN. Surprisingly, nearly $70\%$ of features are from other higher levels. We also use the yellow line to represent large proposals that were assigned to level 4 in FPN. Again, $50\%+$ of the features are pooled from other lower levels. This observation clearly indicates that *features in multiple levels together are helpful for accurate prediction*. It is also a strong support of designing bottom-up path augmentation.

**Adaptive Feature Pooling Structure** *Adaptive feature pooling* is actually simple in implementation and is demonstrated in Figure 1(c). First, for each proposal, we map them to different feature levels, as denoted by dark grey regions in Figure 1(b). Following Mask R-CNN [21], ROIAlign is used to pool feature grids from each level. Then a fusion operation (element-wise max or sum) is utilized to fuse feature grids from different levels.

In following sub-networks, pooled feature grids go through one parameter layer independently, which is followed by the fusion operation, to enable network to adapt features. For example, there are two *fc* layers in the box branch in FPN. We apply the fusion operation after the first layer. Since four consecutive convolutional layers are used in mask prediction branch in Mask R-CNN, we place fusion operation between the first and second convolutional layers. Ablation study is given in Section 4.2. The fused feature grid is used for each proposal for further prediction, *i.e.*, classification, box regression and mask prediction.

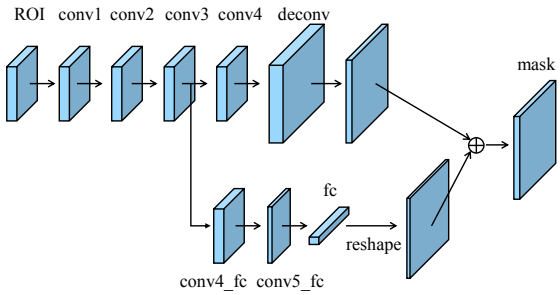


Figure 4. Mask prediction branch with fully-connected fusion.

Our design focuses on fusing information from in-network feature hierarchy instead of those from different feature maps of input image pyramid [52]. It is simpler compared with the process of [4, 62, 31], where L2 normalization, concatenation and dimension reduction are needed.

### 3.3. Fully-connected Fusion

**Motivation** Fully-connected layers, or MLP, were widely used in mask prediction in instance segmentation [10, 41, 34] and mask proposal generation [48, 49]. Results of [8, 33] show that FCN is also competent in predicting pixel-wise masks for instances. Recently, Mask R-CNN [21] applied a tiny FCN on the pooled feature grid to predict corresponding masks avoiding competition between classes.

We note *fc* layers yield different properties compared with FCN where the latter gives prediction at each pixel based on a local receptive field and parameters are shared at different spatial locations. Contrarily, *fc* layers are location sensitive since predictions at different spatial locations are achieved by varying sets of parameters. So they have the ability to adapt to different spatial locations. Also prediction at each spatial location is made with global information of the entire proposal. It is helpful to differentiate instances [48] and recognize separate parts belonging to the same object. Given different properties of *fc* and convolutional layers, we fuse predictions from these two types of layers for better mask prediction.

**Mask Prediction Structure** Our component of mask prediction is light-weighted and easy to implement. The mask branch operates on pooled feature grid for each proposal. As shown in Figure 4, the main path is a small FCN, which consists of 4 consecutive convolutional layers and 1 deconvolutional layer. Each convolutional layer consists of 256 $3 \times 3$ filters and the deconvolutional layer up-samples feature with factor 2. It predicts a binary pixel-wise mask for each class independently to decouple segmentation and classification, similar to that of Mask R-CNN. We further create a short path from layer $conv3$ to a *fc* layer. There are two $3 \times 3$ convolutional layers where the second shrinks channels to half to reduce computational overhead.

A *fc* layer is used to predict a class-agnostic fore-

| Method | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | Backbone |
|---|---|---|---|---|---|---|---|
| Champion 2016 [33] | 37.6 | 59.9 | 40.4 | 17.1 | 41.0 | 56.0 | 6×ResNet-101 |
| Mask R-CNN [21] + FPN [35] | 35.7 | 58.0 | 37.8 | 15.5 | 38.1 | 52.4 | ResNet-101 |
| Mask R-CNN [21] + FPN [35] | 37.1 | 60.0 | 39.4 | 16.9 | 39.9 | 53.5 | ResNeXt-101 |
| PANet / PANet [ms-train] | 36.6 / 38.2 | 58.0 / 60.2 | 39.3 / 41.4 | 16.3 / 19.1 | 38.1 / 41.1 | 53.1 / 52.6 | ResNet-50 |
| PANet / PANet [ms-train] | 40.0 / **42.0** | 62.8 / **65.1** | 43.1 / **45.7** | 18.8 / **22.4** | 42.3 / **44.7** | 57.2 / **58.1** | ResNeXt-101 |

Table 1. Comparison among PANet, winner of COCO 2016 instance segmentation challenge, and Mask R-CNN on COCO *test-dev* subset in terms of Mask AP, where the latter two are baselines.

ground/background mask. It not only is efficient, but also allows parameters in the *fc* layer trained with more samples, leading to better generality. The mask size we use is $28 \times 28$ so that the *fc* layer produces a $784 \times 1 \times 1$ vector. This vector is reshaped to the same spatial size as the mask predicted by FCN. To obtain the final mask prediction, mask of each class from FCN and foreground/background prediction from *fc* are added. Using only one *fc* layer, instead of multiple of them, for final prediction prevents the issue of collapsing the hidden spatial feature map into a short feature vector, which loses spatial information.

## 4. Experiments

We compare our method with state-of-the-arts on challenging COCO [37], Cityscapes [7] and MVD [45] datasets. Our results are top ranked in all of them. Comprehensive ablation study is conducted on the COCO dataset. We also present our results of COCO 2017 Instance Segmentation and Object Detection Challenges.

### 4.1. Implementation Details

We re-implement Mask R-CNN and FPN based on Caffe [29]. All pre-trained models we use in experiments are publicly available. We adopt image centric training [16]. For each image, we sample 512 region-of-interests (ROIs) with positive-to-negative ratio $1 : 3$. Weight decay is $0.0001$ and momentum is set to $0.9$. Other hyper-parameters slightly vary according to datasets and we detail them in respective experiments. Following Mask R-CNN, proposals are from an independently trained RPN [35, 51] for convenient ablation and fair comparison, *i.e.*, the backbone is not shared with object detection and instance segmentation.

### 4.2. Experiments on COCO

**Dataset and Metrics** COCO [37] dataset is among the most challenging ones for instance segmentation and object detection due to the data complexity. It consists of 115k images for training and 5k images for validation (new split of 2017). 20k images are used in *test-dev* and 20k images are used as *test-challenge*. Ground-truth labels of both *test-challenge* and *test-dev* are not publicly available. There are 80 classes with pixel-wise instance mask annotation. We train our models on *train-2017* subset and report results on *val-2017* subset for ablation study. We also report results on *test-dev* for comparison.

We follow the standard evaluation metrics, *i.e.*, AP, $AP_{50}$, $AP_{75}$, $AP_S$, $AP_M$ and $AP_L$. The last three measure performance with respect to objects with different scales. Since our framework is general to both instance segmentation and object detection, we also train independent object detectors. We report mask AP, box ap $AP^{bb}$ of an independently trained object detector, and box ap $AP^{bbM}$ of the object detection branch trained in the multi-task fashion.

**Hyper-parameters** We take 16 images in one image batch for training. The shorter and longer edges of the images are 800 and 1000, if not specially noted. For instance segmentation, we train our model with learning rate $0.02$ for 120k iterations and $0.002$ for another 40k iterations. For object detection, we train one object detector without the mask prediction branch. Object detector is trained for 60k iterations with learning rate $0.02$ and another 20k iterations with learning rate $0.002$. These parameters are adopted from Mask R-CNN and FPN without any fine-tuning.

**Instance Segmentation Results** We report performance of our PANet on *test-dev* for comparison, with and without multi-scale training. As shown in Table 1, our PANet with ResNet-50 trained on multi-scale images and tested on single-scale images already outperforms Mask R-CNN and the champion in 2016, where the latter used larger model ensembles and testing tricks of [23, 33, 10, 15, 39, 62]. Trained and tested with the same image scale 800, our method outperforms the single-model Mask R-CNN with nearly 3 points under the same initial models.

**Object Detection Results** Similar to the way adopted in Mask R-CNN, we also report bounding box results inferred from the box branch. Table 2 shows that our method with ResNet-50, trained and tested on single-scale images, outperforms, by a large margin, all other single-model ones that even used much larger ResNeXt-101 [61] as the initial model. With multi-scale training and single-scale testing, our PANet with ResNet-50 outperforms the champion 2016, which used larger model ensemble and testing tricks.

**Component Ablation Studies** First, we analyze importance of each proposed component. Besides *bottom-up path augmentation*, *adaptive feature pooling* and *fully-connected fusion*, we also analyze *multi-scale training*, *multi-GPU synchronized batch normalization* [67, 28] and *heavier head*. For *multi-scale training*, we set the longer edge to

| Method | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ | $AP^{bb}_{S}$ | $AP^{bb}_{M}$ | $AP^{bb}_{L}$ | Backbone |
|---|---|---|---|---|---|---|---|
| Champion 2016 [27] | 41.6 | 62.3 | 45.6 | 24.0 | 43.9 | 55.2 | 2×ResNet-101 + 3×Inception-ResNet-v2 |
| RentinaNet [36] | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 | ResNet-101 |
| Mask R-CNN [21] + FPN [35] | 38.2 | 60.3 | 41.7 | 20.1 | 41.1 | 50.2 | ResNet-101 |
| Mask R-CNN [21] + FPN [35] | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 | ResNeXt-101 |
| PANet / PANet [ms-train] | 41.2 / 42.5 | 60.4 / 62.3 | 44.4 / 46.4 | 22.7 / 26.3 | 44.0 / 47.0 | 54.6 / 52.3 | ResNet-50 |
| PANet / PANet [ms-train] | 45.0 / **47.4** | 65.0 / **67.2** | 48.6 / **51.8** | 25.4 / **30.1** | 48.6 / **51.7** | 59.1 / **60.0** | ResNeXt-101 |

Table 2. Comparison among PANet, winner of COCO 2016 object detection challenge, RentinaNet and Mask R-CNN on COCO *test-dev* subset in terms of box AP, where the latter three are baselines.

| MRB | RBL | MST | MBN | BPA | AFP | FF | HHD | $AP/AP^{bb}/AP^{bbM}$ | $AP_{50}$ | $AP_{75}$ | $AP_S/AP^{bb}_S/AP^{bbM}_S$ | $AP_M/AP^{bb}_M/AP^{bbM}_M$ | $AP_L/AP^{bb}_L/AP^{bbM}_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | - | - | - | - | - | - | - | 33.6 / 33.9 / - | 55.2 | 35.3 | - / 17.8 / - | - / 37.7 / - | - / 45.8 / - |
| | ✓ | | | | | | | 33.4 / 35.0 / 36.4 | 54.3 | 35.5 | 14.1 / 18.7 / 20.0 | 35.7 / 38.9 / 39.7 | 50.8 / 47.0 / 48.8 |
| | ✓ | ✓ | | | | | | 35.3 / 35.0 / 38.2 | 56.7 | 37.9 | 17.6 / 20.8 / 24.3 | 38.6 / 39.9 / 42.3 | 50.6 / 44.1 / 48.8 |
| | ✓ | ✓ | ✓ | | | | | 35.7 / 37.1 / 38.9 | 57.3 | 38.0 | 18.6 / 24.2 / 25.3 | 39.4 / 42.5 / 43.6 | 51.7 / 47.1 / 49.9 |
| | ✓ | ✓ | ✓ | ✓ | | | | 36.4 / 38.0 / 39.9 | 57.8 | 39.2 | 19.3 / 23.3 / 26.2 | 39.7 / 42.9 / 44.3 | 52.6 / 49.4 / 51.3 |
| | ✓ | ✓ | ✓ | | ✓ | | | 36.3 / 37.9 / 39.6 | 58.0 | 38.9 | 19.0 / 25.4 / 26.4 | 40.1 / 43.1 / 44.9 | 52.4 / 48.6 / 50.5 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | | | 36.9 / 39.0 / 40.6 | 58.5 | 39.7 | 19.6 / 25.7 / **27.0** | 40.7 / 44.2 / 45.7 | 53.2 / 49.5 / 52.1 |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | 37.6 / - / - | 59.1 | 40.6 | **20.3** / - / - | 41.3 / - / - | 53.8 / - / - |
| | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | **37.8 / 39.2 / 42.1** | **59.4** | **41.0** | 19.2 / **25.8 / 27.0** | **41.5 / 44.3 / 47.3** | **54.3 / 50.6 / 54.1** |
| | | | | | | | | **+4.4 / +4.2 / +5.7** | **+5.1** | **+5.5** | **+5.1 / +7.1 / +7.0** | **+5.8 / +5.4 / +7.6** | **+3.5 / +3.6 / +5.3** |

Table 3. Performance in terms of mask AP, box ap $AP^{bb}$ of an independently trained object detector, and box ap $AP^{bbM}$ of the box branch trained with multi-task fashion on *val-2017*. Based on our re-implemented baseline (RBL), we gradually add multi-scale training (MST), multi-GPU synchronized batch normalization (MBN), bottom-up path augmentation (BPA), adaptive feature pooling (AFP), fully-connected fusion (FF) and heavier head (HHD) for ablation study. MRB is short for Mask R-CNN result reported in the original paper. The last row shows total improvement compared with baseline RBL.

$1,400$ and let the other range from $400$ to $1,400$. We calculate mean and variance based on all samples in one batch across all GPUs, do not fix any parameters during training, and make all new layers followed by a batch normalization layer, when using *multi-GPU synchronized batch normalization*. The *heavier head* uses $4$ consecutive $3 \times 3$ convolutional layers shared by box classification and box regression, instead of two *fc* layers. It is similar to the head used in [36]. But the convolutional layers for box classification and box regression branches are not shared in [36].

Our ablation study from the baseline gradually to all components incorporated is conducted on *val-2017* subset and is shown in Table 3. ResNet-50 [23] is our initial model. We report performance in terms of mask AP, box ap $AP^{bb}$ of an independently trained object detector and box ap $AP^{bbM}$ of box branch trained in the multi-task fashion.

**1) Re-implemented Baseline.** Our re-implemented Mask R-CNN performs comparably with the one described in original paper and our object detector performs better.

**2) Multi-scale Training & Multi-GPU Sync. BN.** These two techniques help the network to converge better and increase the generalization ability.

**3) Bottom-up Path Augmentation.** With or without adaptive feature pooling, bottom-up path augmentation consistently improves mask AP and box ap $AP^{bb}$ by more than $0.6$ and $0.9$ respectively. The improvement on big instances is significant, manifesting the usefulness of information sent from lower feature levels.

**4) Adaptive Feature Pooling.** With or without bottom-up path augmentation, adaptive feature pooling consistently improves performance in all scales, which is in accordance with our aforementioned observation that features in other layers are also useful in final prediction.

**5) Fully-connected Fusion.** Fully-connected fusion predicts masks with better quality. It yields $0.7$ improvement in terms of mask AP. It is general for instances at all scales.

**6) Heavier Head.** Heavier head is quite effective for box ap $AP^{bbM}$ of bounding boxes trained in the multi-task fashion. While for mask AP and independently trained object detector, the improvement is smaller.

With all these components in PANet, improvement on mask AP is $4.4$ over baselines. Box ap $AP^{bb}$ of independently trained object detector increases $4.2$. They are significant. Small- and medium-size instances contribute most. Half of the improvement is from *multi-scale training* and *multi-GPU sync. BN*. They are effective strategies.

**Ablation Studies on Adaptive Feature Pooling** Ablation studies on adaptive feature pooling are to verify fusion operation type and location. We place it either between ROIAlign and *fc1*, represented as "fu.*fc1fc2*" or between *fc1* and *fc2*, represented as "*fc1*fu.*fc2*" in Table 4. These settings are also applied to the mask prediction branch. For feature fusing type, max and sum operations are tested.

As shown in Table 4, adaptive feature pooling is not sensitive to the fusion operation type. Allowing a parameter layer to adapt feature grids from different levels, however,

| Settings | AP | $AP_{50}$ | $AP_{75}$ | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ |
|---|---|---|---|---|---|---|
| baseline | 35.7 | 57.3 | 38 | 37.1 | 58.9 | 40.0 |
| fu.*fc1fc2* | 35.7 | 57.2 | 38.2 | 37.3 | 59.1 | 40.1 |
| *fc1*fu.*fc2* | **36.3** | **58.0** | **38.9** | **37.9** | **60.0** | **40.7** |
| MAX | **36.3** | **58.0** | **38.9** | 37.9 | **60.0** | **40.7** |
| SUM | 36.2 | **58.0** | 38.8 | **38.0** | 59.8 | **40.7** |

Table 4. Ablation study on adaptive feature pooling on *val-2017* in terms of mask AP and box ap $AP^{bb}$ of the independently trained object detector.

| Settings | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| baseline | 36.9 | 58.5 | 39.7 | 19.6 | 40.7 | 53.2 |
| conv2 | 37.5 | **59.3** | 40.1 | **20.7** | 41.2 | **54.1** |
| conv3 | **37.6** | 59.1 | **40.6** | 20.3 | **41.3** | 53.8 |
| conv4 | 37.2 | 58.9 | 40.0 | 19.0 | 41.2 | 52.8 |
| PROD | 36.9 | 58.6 | 39.7 | 20.2 | 40.8 | 52.2 |
| SUM | **37.6** | **59.1** | **40.6** | **20.3** | **41.3** | **53.8** |
| MAX | 37.1 | 58.7 | 39.9 | 19.9 | 41.1 | 52.5 |

Table 5. Ablation study on fully-connected fusion on *val-2017* in terms of mask AP.

| | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| Champion 2015 [10] | 28.4 | 51.6 | 28.1 | 9.4 | 30.6 | 45.6 |
| Champion 2016 [33] | 37.6 | 59.9 | 40.4 | 17.1 | 41.0 | 56.0 |
| Our Team 2017 | **46.7** | **69.5** | **51.3** | **26.0** | **49.1** | **64.0** |
| PANet baseline | 38.2 | 60.2 | 41.4 | 19.1 | 41.1 | 52.6 |
| + DCN [11] | 39.5 | 62.0 | 42.8 | 19.8 | 42.2 | 54.7 |
| + testing tricks | 42.0 | 63.5 | 46.0 | 21.8 | 44.4 | 58.1 |
| + larger model | 44.4 | 67.0 | 48.5 | 23.6 | 46.5 | 62.2 |
| + ensemble | **46.7** | **69.5** | **51.3** | **26.0** | **49.1** | **64.0** |

Table 6. Mask AP of COCO Instance Segmentation Challenge in different years on *test-dev*.

is of greater importance. In our final system, we use max fusion operation behind the first parameter layer.

**Ablation Studies on Fully-connected Fusion** We investigate performance when instantiating the augmented *fc* branch differently. We consider two aspects, *i.e.*, the layer to start the new branch and the way to fuse predictions from the new branch and FCN. We experiment with creating new paths from $conv2$, $conv3$ and $conv4$, respectively. "max", "sum" and "product" operations are used for fusion. We take our reimplemented Mask R-CNN with bottom-up path augmentation and adaptive feature pooling as the baseline. Corresponding results are listed in Table 5. They clearly show that staring from $conv3$ and taking sum for fusion produce the best results.

**COCO 2017 Challenge** With PANet, we participated in the COCO 2017 Instance Segmentation and Object Detection Challenges. Our framework reaches the $1^{st}$ place in Instance Segmentation task and the $2^{nd}$ place in Object Detection task without large-batch training. As shown in Tables 6 and 7, compared with last year champion, we achieve $9.1\%$ absolute and $24\%$ relative improvement on instance segmentation. While for object detection, $9.4\%$ absolute and $23\%$ relative improvement is yielded.

| | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ | $AP^{bb}_S$ | $AP^{bb}_M$ | $AP^{bb}_L$ |
|---|---|---|---|---|---|---|
| Champion 2015 [23] | 37.4 | 59.0 | 40.2 | 18.3 | 41.7 | 52.9 |
| Champion 2016 [27] | 41.6 | 62.3 | 45.6 | 24.0 | 43.9 | 55.2 |
| Our Team 2017 | **51.0** | **70.5** | **55.8** | **32.6** | **53.9** | **64.8** |

Table 7. Box AP of COCO Object Detection Challenge in different years on *test-dev*.

There are a few more details for the top performance. First, we use deformable convolution where DCN [11] is adopted. The common testing tricks [23, 33, 10, 15, 39, 62], such as multi-scale testing, horizontal flip testing, mask voting and box voting, are used too. For multi-scale testing, we set the longer edge to $1,400$ and let the other range from $600$ to $1,200$ with step $200$. Only $4$ scales are used. Second, we use larger initial models from publicly available ones. We use 3 ResNeXt-101 ($64 \times 4d$) [61], 2 SE-ResNeXt-101 ($32 \times 4d$) [25], 1 ResNet-269 [64] and 1 SENet [25] as ensemble for bounding box and mask generation. Performance with different larger initial models are similar. One ResNeXt-101 ($64 \times 4d$) is used as the base model to generate proposals. We train these models with different random seeds, with and without balanced sampling [55] to enhance diversity between models. Detection results are acquired by tightening instance masks. We show a few visual results in Figure 5 – most of our predictions are with high quality.

### 4.3. Experiments on Cityscapes

**Dataset and Metrics** Cityscapes [7] contains street scenes captured by car-mounted cameras. There are $2,975$ training images, $500$ validation and $1,525$ testing images with fine annotations. Another 20k images are with coarse annotations, excluded for training. We report our results on *val* and secret *test* subsets. $8$ semantic classes are annotated with instance masks. Each image is with size $1024 \times 2048$. We evaluate results in terms of AP and $AP_{50}$.

**Hyper-parameters** We use the same set of hyper-parameters as in Mask R-CNN [21] for fair comparison. Specifically, we use images with shorter edge randomly sampled from $\{800, 1024\}$ for training and use images with shorter edge $1024$ for inference. No testing tricks or DCN is used. We train our model for 18k iterations with learning rate $0.01$ and another 6k iterations with learning rate $0.001$. $8$ images ($1$ image per GPU) are in one image batch. ResNet-50 is taken as the initial model on this dataset.

**Results and Ablation Study** We compare with state-of-the-arts on *test* subset in Table 8. Trained on "fine-only" data, our method outperforms Mask R-CNN with "fine-only" data by $5.6$ points. It is even comparable with Mask R-CNN pre-trained on COCO. By pre-training on COCO, we outperform Mask R-CNN with the same setting by $4.4$ points. Visual results are shown in Figure 5.

Our ablation study to analyze the improvement on *val* subset is given in Table 9. Based on our re-implemented

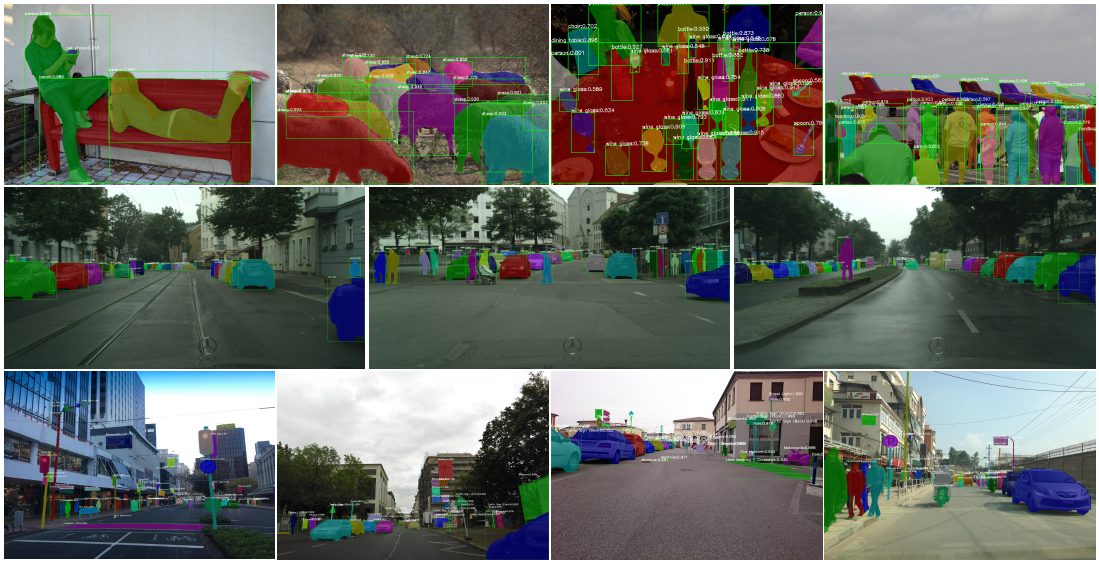

Figure 5. Visual results of our method on COCO *test-dev*, Cityscapes *test* and MVD *test* respectively in the three rows.

| Methods | AP [val] | AP | $AP_{50}$ | person | rider | car | truck | bus | train | motorcycle | bicycle |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SGN [38] | 29.2 | 25.0 | 44.9 | 21.8 | 20.1 | 39.4 | 24.8 | 33.2 | 30.8 | 17.7 | 12.4 |
| Mask R-CNN [fine-only] [21] | 31.5 | 26.2 | 49.9 | 30.5 | 23.7 | 46.9 | 22.8 | 32.2 | 18.6 | 19.1 | 16.0 |
| SegNet | - | 29.5 | 55.6 | 29.9 | 23.4 | 43.4 | 29.8 | 41.0 | **33.3** | 18.7 | 16.7 |
| Mask R-CNN [COCO] [21] | 36.4 | 32.0 | 58.1 | 34.8 | 27.0 | 49.1 | 30.1 | 40.9 | 30.9 | 24.1 | 18.7 |
| PANet [fine-only] | 36.5 | 31.8 | 57.1 | 36.8 | 30.4 | 54.8 | 27.0 | 36.3 | 25.5 | 22.6 | 20.8 |
| PANet [COCO] | **41.4** | **36.4** | **63.1** | **41.5** | **33.6** | **58.2** | **31.8** | **45.3** | 28.7 | **28.2** | **24.1** |

Table 8. Results on Cityscapes *val* subset, denoted as AP [val], and on Cityscapes *test* subset, denoted as AP.

| Methods | AP | $AP_{50}$ |
|---|---|---|
| our re-implement | 33.1 | 59.1 |
| our re-implement + MBN | 34.6 | 62.4 |
| PANet | **36.5** | **62.9** |

Table 9. Ablation study results on Cityscapes *val* subset. Only fine annotations are used for training. MBN is short for multi-GPU synchronized batch normalization.

| Methods | AP [test] | $AP_{50}$ [test] | AP [val] | $AP_{50}$ [val] |
|---|---|---|---|---|
| UCenter-Single [40] | - | - | 22.8 | 42.5 |
| UCenter-Ensemble [40] | 24.8 | 44.2 | 23.7 | 43.5 |
| PANet | - | - | 23.6 | 43.3 |
| PANet [test tricks] | **26.3** | **45.8** | **24.9** | **44.7** |

Table 10. Results on MVD *val* subset and *test* subset.

baseline, we add multi-GPU synchronized batch normalization to help network converge better. It improves the accuracy by 1.5 points. With our full PANet, the performance is further boosted by 1.9 points.

### 4.4. Experiments on MVD

MVD [45] is a relatively new and large-scale dataset for instance segmentation. It provides $25,000$ images on street scenes with fine instance-level annotations for 37 semantic classes. They are captured from several countries using different devices. The content and resolution vary greatly. We train our model on *train* subset with ResNet-50 as initial model and report performance on *val* and secret *test* subsets in terms of AP and $AP_{50}$.

We present our results in Table 10. Compared with UCenter [40] – winner on this dataset in LSUN 2017 instance segmentation challenge, our PANet with one ResNet-50 tested on single-scale images already performs comparably with the ensemble result with pre-training on COCO. With multi-scale and horizontal flip testing, which are also adopted by UCenter, our method performs better. Qualitative results are illustrated in Figure 5.

## 5. Conclusion

We have presented our PANet for instance segmentation. We designed several simple and yet effective components to enhance information propagation in representative pipelines. We pool features from all feature levels and shorten the distance among lower and topmost feature levels for reliable information passing. Complementary path is augmented to enrich feature for each proposal. Impressive results are produced. Our future work will be to extend our method to videos and RGBD data.

## Acknowledgements

We would like to thank Yuanhao Zhu, Congliang Xu and Qingping Fu in SenseTime for the technical support.

## References


[1] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2

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

[3] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017. 1, 2

[4] S. Bell, C. L. Zitnick, K. Bala, and R. B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 2, 4

[5] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016. 1, 2

[6] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *arXiv:1707.01629*, 2017. 1

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 5, 7

[8] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 2, 4

[9] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2

[10] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *CVPR*, 2016. 2, 4, 5, 7

[11] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 7

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 2010. 1

[13] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *arXiv:1701.06659*, 2017. 1, 2

[14] G. Ghiasi and C. C. Fowlkes. Laplacian reconstruction and refinement for semantic segmentation. In *ECCV*, 2016. 1, 2

[15] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware CNN model. In *ICCV*, 2015. 3, 5, 7

[16] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 5

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

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*. 2014. 2

[19] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2

[20] Z. Hayder, X. He, and M. Salzmann. Boundary-aware instance segmentation. In *CVPR*, 2017. 2

[21] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 4, 5, 6, 7, 8

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

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 5, 6, 7

[24] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1

[25] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv:1709.01507*, 2017. 7

[26] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1

[27] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 2, 6, 7

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *MM*, 2014. 5

[30] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: From edges to instances with multicut. In *CVPR*, 2017. 2

[31] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016. 1, 2, 4

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 3

[33] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 1, 2, 4, 5, 7

[34] X. Liang, Y. Wei, X. Shen, Z. Jie, J. Feng, L. Lin, and S. Yan. Reversible recursive instance-level object segmentation. In *CVPR*, 2016. 2, 4

[35] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2, 3, 5, 6

[36] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *ICCV*, 2017. 6

[37] 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. 1, 5

[38] S. Liu, J. Jia, S. Fidler, and R. Urtasun. SGN: Sequential grouping networks for instance segmentation. In *ICCV*, 2017. 1, 2, 8

[39] S. Liu, C. Lu, and J. Jia. Box aggregation for proposal decimation: Last mile of object detection. In *ICCV*, 2015. 5, 7

[40] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. LSUN'17: insatnce segmentation task, UCenter winner team. https://research.mapillary.com/img/lsun/lsun17_scene_parsing_winners.pptx, 2017. 8

[41] S. Liu, X. Qi, J. Shi, H. Zhang, and J. Jia. Multi-scale patch aggregation (MPA) for simultaneous detection and segmentation. *CVPR*, 2016. 2, 4

[42] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 1, 2

[43] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015. 3

[44] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[45] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 1, 5, 8

[46] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1, 2

[47] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters - improve semantic segmentation by global convolutional network. In *CVPR*, 2017. 2, 3

[48] P. H. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, 2015. 2, 4

[49] P. H. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 2, 4

[50] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017. 2

[51] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 5

[52] S. Ren, K. He, R. B. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *PAMI*, 2017. 3, 4

[53] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, 2016. 2

[54] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[55] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 7

[56] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv:1612.06851*, 2016. 2

[57] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 3

[58] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 2

[59] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *GCPR*, 2016. 2

[60] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[61] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 5, 7

[62] S. Zagoruyko, A. Lerer, T. Lin, P. H. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. In *BMVC*, 2016. 2, 3, 4, 5, 7

[63] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*. 2014. 3

[64] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, H. Zhou, and X. Wang. Crafting GBD-Net for object detection. *arXiv:1610.02579*, 2016. 3, 7

[65] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected MRFs. In *CVPR*, 2016. 2

[66] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with CNNs. In *ICCV*, 2015. 2

[67] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 3, 5

[68] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2