BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation

Hao Chen[1]*, Kunyang Sun[2,1]*, Zhi Tian[1], Chunhua Shen[1], Yongming Huang[2], Youliang Yan[3]
[1] The University of Adelaide, Australia [2] Southeast University, China [3] Huawei Noah's Ark Lab
{hao.chen01, zhi.tian, chunhua.shen}@adelaide.edu.au {sunky, huangym}@seu.edu.cn yanyouliang@huawei.com

Abstract

Instance segmentation is one of the fundamental vision tasks. Recently, fully convolutional instance segmentation methods have drawn much attention as they are often simpler and more efficient than two-stage approaches like Mask R-CNN. To date, almost all such approaches fall behind the two-stage Mask R-CNN method in mask precision when models have similar computation complexity, leaving great room for improvement. In this work, we achieve improved mask prediction by effectively combining instance-level information with semantic information with lower-level fine-granularity. Our main contribution is a blender module which draws inspiration from both top-down and bottom-up instance segmentation approaches. The proposed BlendMask can effectively predict dense per-pixel position-sensitive instance features with very few channels, and learn attention maps for each instance with merely one convolution layer, thus being fast in inference. BlendMask can be easily incorporate with the state-of-the-art one-stage detection frameworks and outperforms Mask R-CNN under the same training schedule while being faster. A light-weight version of BlendMask achieves 36.0 mAP at 27 FPS evaluated on a single 1080Ti. Because of its simplicity and efficacy, we hope that our BlendMask could serve as a simple yet strong baseline for a wide range of instance-wise prediction tasks.

1. Introduction

The top performing object detectors and segmenters often follow a two-stage paradigm. They consist of a fully convolutional network, region proposal network (RPN), to perform dense prediction of the most likely regions of interest (RoIs). A set of light-weight networks, a.k.a. heads, are applied to re-align the features of RoIs and generate predictions [22]. The quality and speed for mask generation is strongly tied to the structure of the mask heads. In addition, it is difficult for independent heads to share features with related tasks such as semantic segmentation which causes trouble for network architecture optimization.

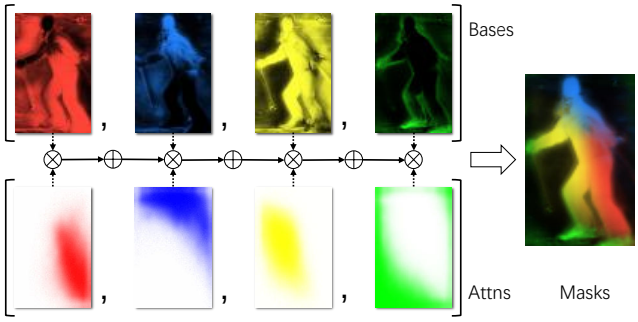


Figure 1: Blending process. We illustrate an example of the learned bases and attentions. Four bases and attention maps are shown in different colors. The first row are the bases, and the second row are the attentions. Here $\otimes$ represents element-wise product and $\oplus$ is element-wise sum. Each basis multiplies its attention and then is summed to get the final mask.

Recent advances in one-stage object detection prove that one-stage methods such as FCOS can outperform their two-stage counterparts in accuracy [23]. Enabling such one-stage detection frameworks to perform dense instance segmentation is highly desirable as 1) models consisting of only conventional operations are simpler and easier for cross-platform deployment; 2) a unified framework provides convenience and flexibility for multi-task network architecture optimization.

Dense instance segmenters can date back to DeepMask [21], a top-down approach which generates dense instance masks with a sliding window. The representation of mask is encoded into a one-dimensional vector at each spatial location. Albeit being simple in structure, it has several obstacles in training that prevent it from achieving superior performance: 1) local-coherence between features and masks is lost; 2) the feature representation is redundant because a mask is repeatedly encoded at each foreground feature; 3) position information is degraded after downsampling with strided convolutions.

The first issue was studied by Dai *et al*. [8], who attempt to retain local-coherence by keeping multiple position-sensitive maps. This idea has been explored to its limits by

*Equal contribution. This work was done when K. Sun was visiting The University of Adelaide. Correspondence should be addressed to C. Shen and Y. Huang.

Chen *et al*. [7], who proposes a dense aligned representation for each location of the target instance mask. However, this approach trades representation efficiency for alignment, making the second issue difficult to resolve. The third issue prevents heavily downsampled features to provide detailed instance information.

Recognizing these difficulties, a line of research takes a bottom-up strategy [1, 19, 20]. These methods generate dense per-pixel embedding features and use some techniques to group them. Grouping strategies vary from simple clustering [4] to graph-based algorithms [19] depending on the embedding characteristics. By performing per-pixel predictions, the local-coherence and position information is well retained. The shortcomings for bottom-up approaches are: 1) heavy reliance on the dense prediction quality, leading to sub-par performance and fragmented/joint masks; 2) limited generalization ability to complex scenes with a large number of classes; 3) requirement for complex post-processing techniques.

In this work, we consider hybridizing top-down and bottom-up approaches. We recognize two important predecessors, FCIS [16] and YOLACT [3]. They predict instance-level information such as bounding box locations and combine it with per-pixel predictions using cropping (FCIS) and weighted summation (YOLACT), respectively. We argue that *these overly simplified assembling designs may not provide a good balance for the representation power of top- and bottom-level features.*

Higher-level features correspond to larger receptive field and can better capture overall information about instances such as poses, while lower-level features preserve better location information and can provide finer details. One of the focuses of our work is to investigate ways to better merging these two in fully convolutional instance segmentation. More specifically, we generalize the operations for proposal-based mask combination by enriching the instance-level information and performing more fine-grained position-sensitive mask prediction. We carry out extensive ablation studies to discover the optimal dimensions, resolutions, alignment methods, and feature locations. Concretely, we are able to achieve the followings:

- We devise a flexible method for proposal-based instance mask predictor called blender, which incorporates rich instance-level information with accurate dense pixel features. It can be added to most object detectors with modest computation overhead. In head-to-head comparison, our blender surpasses the merging techniques in YOLACT [3] and FCIS [16] by 1.9 and 1.3 points in mAP on the COCO dataset respectively.
- One obvious advantage of BlendMask is that its inference time does not increase with the number of predictions as conventional two-stage methods do, which makes it more robust in real-time scenarios.
- Compared with Mask R-CNN's mask head, which is typically of 28×28 resolution, BlendMask's the bottom module is able to output masks of much higher resolution, due to its flexibility and the bottom module not being strictly tied to the FPN. Thus BlendMask is able to produce masks with more accurate edges, as shown in Figure 4. For applications such as graphics, this can be very important.
- The performance of BlendMask achieves mAP of 37.0% with the ResNet-50 [14] backbone and 41.3% mAP with ResNet-101 on the COCO dataset, outperforming Mask R-CNN [12] in accuracy while being faster. We set new records for fully convolutional instance segmentation, surpassing TensorMask [7] by 2.3 points in mask mAP with only half training iterations and $1/5$ inference time.

2. Related work

Anchor-free object detection Recent advances in object detection unveil the possibilities of removing bounding box anchors [23], largely simplifying the detection pipeline. This much simpler design improves the box average precision (AP^{bb}) by 2.7% comparing to its anchor-based counterpart RetinaNet [17]. One possible reason responsible for the improvement is that without the restrictions of predefined anchor shapes, targets are freely matched to prediction features according to their effective receptive field. The hints for us are twofold. First, it is important to map target sizes with proper pyramid levels to fit the effective receptive field for the features. Second, *removing anchors enables us to assign heavier duties to the top-level instance prediction module without introducing overall computation overhead*. For example, inferring shape and pose information alongside the bounding box detection would take about eight times more computation for anchor-based frameworks than ours. This makes it intractable for anchor based detectors to balance the top *vs.* bottom workload (i.e., learning instance-aware maps[1] *vs.* bases). We assume that this might be the reason why YOLACT can only learn one single scalar coefficient for each prototype/basis given an instance when computation complexity is taken into account. *Only with the use of anchor-free bounding box detectors, this restriction is removed.*

Detect-then-segment instance segmentation The dominant instance segmentation paradigms take the two-stage methodology, first detecting the objects and then predicting the foreground masks on each of the proposals. The success of this framework partially is due to the alignment operation, RoIAlign [12], which provides local-coherence for the second-stage RoI heads missing in all one-stage top-down

[1]Attention maps for BlendMask and simple weight scalars for YOLACT.

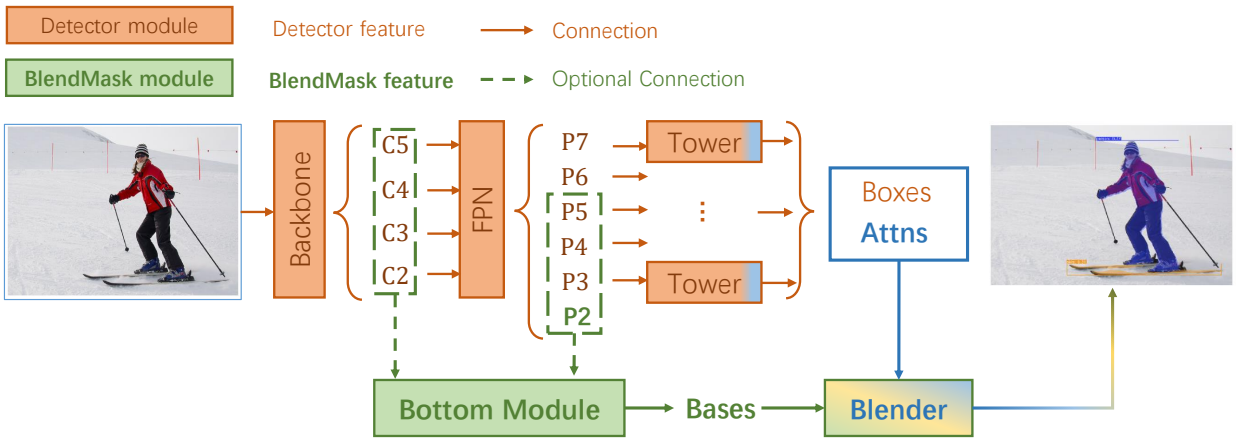


Figure 2: BlendMask pipeline Our framework builds upon the state-of-the-art FCOS object detector [23] with minimal modification. The bottom module uses either backbone or FPN features to predict a set of bases. A single convolution layer is added on top of the detection towers to produce attention masks along with each bounding box prediction. For each predicted instance, the blender crops the bases with its bounding box and linearly combine them according the learned attention maps. Note that the Bottom Module can take features either from 'C', or 'P' as the input.

approaches. However, two issues exist in two-stage frameworks. For complicated scenarios with many instances, inference time for two-stage methods is proportional to the number of instances. Furthermore, the resolution for the RoI features and resulting mask is limited. We discuss the second issue in detail in Section 4.3.

These problems can be partly solved by replacing a RoI head with a simple crop-and-assemble module. In FCIS, Li *et al*. [16] add a bottom module to a detection network, for predicting position-sensitive score maps shared by all instances. This technique was first used in R-FCN [9] and later improved in MaskLab [5]. Each channel of the k^2 score maps corresponds to one crop of $k \times k$ evenly partitioned grid tiles of the proposal. Each score map represents the likelihood of the pixel belongs to a object and is at a certain relative position. Naturally, a higher resolution for location crops leads to more accurate predictions, but the computation cost also increases quadratically. Moreover, there are special cases where FCIS representation is not sufficient. When two instances share center positions (or any other relative positions), the score map representation on that crop is ambiguous, it is impossible to tell which instance this crop is describing.

In YOLACT [3], an improved approach is used. Instead of using position-controlled tiles, a set of mask coefficients are learned alongside the box predictions. Then this set of coefficients guides the linear combination of cropped bottom mask bases to generate the final mask. Comparing to FCIS, the responsibility for predicting instance-level information is assigned to the top-level. We argue that using scalar coefficients to encode the instance information is suboptimal.

To break through these limitations, we propose a new proposal-based mask generation framework, termed BlendMask. The top- and bottom-level representation workloads are balanced by a blender module. Both levels are guaranteed to describe the instance information within their best capacities. As shown in our experiments in Section 4, our blender module improves the performance of bases combination methods comparing to YOLACT and FCIS by a large margin without increasing computation complexity.

Refining coarse masks with lower-level features BlendMask merges top-level coarse instance information with lower-level fine-granularity. This idea resembles MaskLab [5] and Instance Mask Projection (IMP) [10], which concatenates mask predictions with lower layers of backbone features. The differences are clear. Our coarse mask acts like an attention map. The generation is extremely light-weight, without the need of using semantic or positional supervision, and is closely tied to the object generation. As shown in Section 3.4, our lower-level features have clear contextual meanings, even though not explicitly guided by bins or crops. Further, our blender does not require a subnet on top of the merged features as in MaskLab [5] and IMP [10], which makes our method more efficient.

3. BlendMask

3.1. Overall pipeline

BlendMask consists of a detector network and a mask branch. The mask branch has three parts, a bottom module to predict the score maps, a top layer to predict the instance attentions, and a blender module to merge the scores with attentions. The whole network is illustrated in Figure 2.

Bottom module Similar to other proposal-based fully convolutional methods [3, 16], we add a bottom module predicting score maps which we call bases, $\mathbf{B}$. $\mathbf{B}$ has a shape of

$N \times K \times \frac{H}{s} \times \frac{W}{s}$, where N is the batch size, K is the number of bases, $H \times W$ is the input size and s is the score map output stride. We use the decoder of DeepLabV3+ in our experiments. Other dense prediction modules should also work without much difference. The input for the bottom module could be backbone features like conventional semantic segmentation networks [6], or the feature pyramids like YOLACT and Panoptic FPN [15].

Top layer We also append a single convolution layer on each of the detection towers to predict top-level attentions $\mathbf{A}$. Unlike the mask coefficients in YOLACT, which for each pyramid with resolution $W_l \times H_l$ takes the shape of $N \times K \times H_l \times W_l$, our $\mathbf{A}$ is a tensor at each location with shape $N \times (K \cdot M \cdot M) \times H_l \times W_l$, where $M \times M$ is the attention resolution. With its 3D structure, our attention map can encode instance-level information, e.g. the coarse shape and pose of the object. M is typically smaller[2] than the mask predictions in top-down methods since we only ask for a rough estimate. We predict it with a convolution with $K \cdot M \cdot M$ output channels. Before sending them into the next module, we first apply FCOS [23] post-process to select the top D box predictions $P = \{\mathbf{p}_d \in \mathbb{R}^4_{\geq 0} | d = 1 \dots D\}$ and corresponding attentions $A = \{\mathbf{a}_d \in \mathbb{R}^{K \times M \times M} | d = 1 \dots D\}$.

Blender module is the key part of our BlendMask. It combines position-sensitive bases according to the attentions to generate the final prediction. We discuss this module in detail in the next section.

3.2. Blender module

The inputs of the blender module are bottom-level bases $\mathbf{B}$, the selected top-level attentions A and bounding box proposals P. First we use RoIPooler in Mask R-CNN [12] to crop bases with each proposal $\mathbf{p}_d$ and then resize the region to a fixed size $R \times R$ feature map $\mathbf{r}_d$.

$$\mathbf{r}_d = \text{RoIPool}_{R \times R}(\mathbf{B}, \mathbf{p}_d), \quad \forall d \in \{1 \dots D\}. \tag{1}$$

More specifically, we use sampling ratio 1 for RoIAlign, i.e. one bin for each sampling point. The performance of using nearest and bilinear poolers are compared in Table 6. During training, we simply use ground truth boxes as the proposals. During inference, we use bbox predictions.

We interpolate $\mathbf{a}_d$ from $M \times M$ to $R \times R$, into the shapes of $R = \{\mathbf{r}_d | d = 1 \dots D\}$.

$$\mathbf{a}'_d = \text{interpolate}_{M \times M \to R \times R}(\mathbf{a}_d), \quad \forall d \in \{1 \dots D\}. \tag{2}$$

Then $\mathbf{a}'_d$ is normalize with softmax function along the K dimension to make it a set of score maps $\mathbf{s}_d$.

$$\mathbf{s}_d = \text{softmax}(\mathbf{a}'_d), \quad \forall d \in \{1 \dots D\}. \tag{3}$$

[2]The largest M we try is 14.

Then we apply element-wise product between each entity $\mathbf{r}_d$, $\mathbf{s}_d$ of the regions R and scores S, and sum along the K dimension to get our mask logit $\mathbf{m}_d$:

$$\mathbf{m}_d = \sum_{k=1}^{K} \mathbf{s}_d^k \circ \mathbf{r}_d^k, \quad \forall d \in \{1 \dots D\}, \tag{4}$$

where k is the index of the basis. We visualize the mask blending process with $K = 4$ in Figure 1.

3.3. Configurations and baselines

We consider the following configurable hyper-parameters for BlendMask:

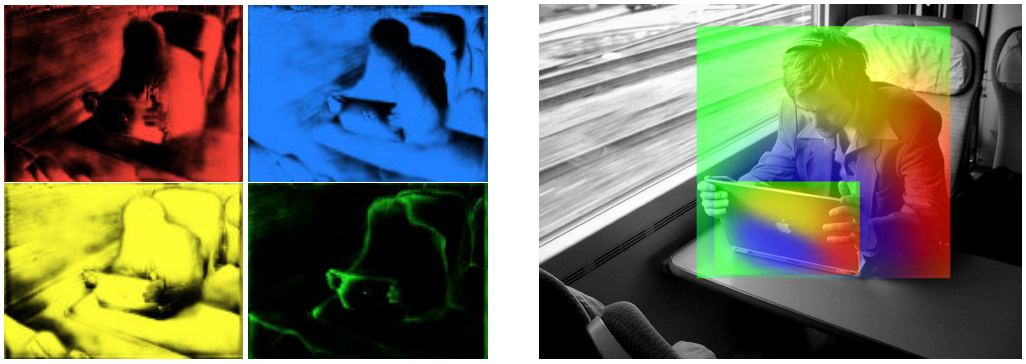
- R, the bottom-level RoI resolution,
- M, the top-level prediction resolution,
- K, the number of bases,
- bottom module input features, it can either be features from the backbone or the FPN,
- sampling method for bottom bases, nearest-neighbour or bilinear pooling,
- interpolation method for top-level attentions, nearest neighbour or bilinear upsampling.

We represent our models with abbreviation `R_K_M`. For example, `28_4_4` represents bottom-level region resolution of 28×28, 4 number of bases and 4×4 top-level instance attentions. By default, we use backbone features C3 and C5 to keep aligned with DeepLabv3+ [6]. Nearest neighbour interpolation is used in top-level interpolation, for a fair comparison with FCIS [16]. Bilinear sampling is used in the bottom level, consistent with RoIAlign [12].

3.4. Semantics encoded in learned features

By examining the generated bases and attentions on `val2017`, we observe this pattern. On its bases, BlendMask encodes two types of local information, 1) whether the pixel is on an object (semantic), 2) whether the pixel is on certain part of the object (position-sensitive).

The complete bases and attentions projected onto the original image are illustrated in Figure 3. The first two bases (red and blue) detects points on the upper-right and bottom-left parts of the objects. The third (yellow) base activates on points more likely to be on an object. The fourth (green) base only activates on the borders of objects. Position-sensitive features help us separate overlapping instances, which enables BlendMask to represent all instances more efficiently than YOLACT [3]. The positive semantic mask makes our final prediction smoother than FCIS [16] and the negative one can further suppress out-of-instance activations. We compare our blender with YOLACT and FCIS counterparts in Table 1. BlendMask can learn more accurate features than YOLACT and FCIS with much fewer number of bases (4 *vs.* 32 *vs.* 49, see Section 4.2).



(a) Bottom-Level Bases **(b)** Top-Level attentions

Figure 3: Detailed view of learned bases and attentions. The left four images are the bottom-level bases. The right image is the top-level attentions. Colors on each position of the attentions correspond to the weights of the bases, indicating from which part of which base is the mask assembled.

4. Experiments

Our experiments are reported on the MSCOCO 2017 instance segmentation datatset [18]. It contains 123K images with 80-class instance labels. Our models are trained on the `train2017` split (115K images) and the ablation study is carried out on the `val2017` split (5K images). Final results are on `test-dev`. The evaluation metrics are COCO mask average precision (AP), AP at IoU 0.5 (AP_{50}), 0.75 (AP_{75}) and AP for objects at different sizes AP_S, AP_M, and AP_L.

Training details Unless specified, ImageNet pre-trained ResNet-50 [13] is used as our backbone network. DeepLabv3+ [6] decoder with channel width 128 is used as our bottom module. For ablation study, all the networks are trained with the $1\times$ schedule of FCOS [23], i.e., 90K iterations, batch size 16 on 4 GPUs, and base learning rate 0.01 with constant warm-up of 1k iterations. The learning rate is reduced by a factor of 10 at iteration 60K and 80K. All hyperparameters are set to be the same with FCOS [23].

Testing details The unit for inference time is 'ms' in all our tables. For the ablation experiments, performance and time of our models are measured with one image per batch on one 1080Ti GPU.

4.1. Ablation experiments

We investigate the effectiveness of our blender module by carrying out ablation experiments on the configurable hyperparameters in Section 3.3.

Merging methods: Blender *vs.* **YOLACT** *vs.* **FCIS** Similar to our method, YOLACT [3] and FCIS [16] both merge proposal-based bottom regions to create mask prediction. YOLACT simply performs a weighted sum of the channels of the bottom regions; FCIS assembles crops of position-sensitive masks without modifications. Our blender can be regarded as a generalization where both YOLACT and FCIS merging are special cases: The blender with 1×1 top-level resolution degenerates to YOLACT; and FCIS is the case where we use fixed one-hot blending attentions and nearest neighbour top-level interpolation.

Method	AP	AP_{50}	AP_{75}
Weighted-sum	29.7	52.2	30.1
Assembler	30.3	52.5	31.3
Blender	**31.6**	**53.4**	**33.3**

Table 1: Comparison of different strategies for merging top and bottom modules. Here the model used is `28_4_4`. Weighted-sum is our analogy to YOLACT, reducing the top resolution to 1×1. Assembler is our analogy to FCIS, where the number of bases is increased to 16, matching each of the region crops without the need of top-level attentions.

R	M	Time (ms)	AP	AP_S	AP_M	AP_L
28	2	**72.7**	30.6	14.3	34.1	42.5
	4	72.9	31.6	14.8	35.2	45.0
	7	73.9	32.0	15.3	35.6	45.0
56	4	72.9	32.5	14.9	36.1	46.0
	7	74.1	33.1	15.1	36.6	**47.7**
	14	77.7	**33.3**	**16.3**	**36.8**	47.4

Table 2: Resolutions: Performance by varying top-/bottom-level resolutions, with the number of bases $K = 4$ for all models. Top-level attentions are interpolated with nearest neighbour. Bottom module uses backbone features C3, C5. The performance increases as the attention resolution grows, saturating at resolutions of near $1/4$ of the region sizes.

Results are shown in Table 1. Our blender surpasses the other alternatives by a large margin. We assume the reason is that other methods lack instance-aware guidance on the top. By contrast, our blender has a fine-grained top-level attention map, as illustrated in Figure 3.

Top and bottom resolutions: We measure the performances of our model with different top- and bottom-level resolutions, trying bottom pooler resolution R being 28 and 56, with R/M ratio from 14 to 4. As shown in Table 2, by increasing the attention resolution, we can incorporate more detailed instance-level information while keeping the running time roughly the same. Notice that the gain slows down at higher resolutions revealing limit of detailed in-

formation on the top-level. So we don't include larger top settings with R/M ratio smaller than 4.

Different from two-stage approaches, increasing the bottom-level bases pooling resolution does not introduce much computation overhead. Increasing it from 28 to 56 only increases the inference time within 0.2ms while mask AP increases by 1 point. In further ablation experiment, we set $R = 56$ and $M = 7$ for our baseline models.

Number of bases: YOLACT [3] uses 32 bases concerning the inference time. With our blender, the number of bases can be further reduced, to even just one. We report our models with number of bases varying from 1 to 8. Different from normal blender, the one-basis version uses sigmoid activation on both the base and the attention map. Results are shown in Table 3. Since instance-level information is better represented with the top-level attentions, we only need 4 bases to get the optimal accuracy. $K = 4$ is adopted by all subsequent experiments.

Bottom feature locations: backbone *vs.* **FPN** By using FPN features, we can improve the performance while reducing the running time (see Table 4). In later experiments, if not specified, we use P3 and P5 of FPN as our bottom module input.

Interpolation method: nearest *vs.* **bilinear** In Mask R-CNN [12], RoIAlign plays a crucial role in aligning the pooled features to keep local-coherence. We investigate the effectiveness of bilinear interpolation for bottom RoI sampling and top-level attention re-scaling. As shown in Table 5, changing top interpolation from nearest to bilinear yields a marginal improvement of 0.2 AP.

The results of bottom sampling with RoIPool [11] (nearest) and RoIAlign [12] (bilinear) are shown in Table 6. For both resolutions, the aligned bilinear sampling could improve the performance by almost 2AP. Using aligned features for the bottom-level is more crucial, since it is where the detailed positions are predicted. Bilinear top and bottom interpolation are adopted for our final models.

Other improvements: We experiment on other tricks to improve the performance. First we add auxiliary semantic segmentation supervision on P3 similar to YOLACT [3]. Then we increase the width of our bottom module from 128 to 256. Finally, we reduce the bases output stride from 8 to 4, to produce higher-quality bases. We achieve this by using P2 and P5 as the bottom module input. Table 7 shows the results. By adding semantic loss, detection and segmentation results are both improved. This is an interesting effect since the instance segmentation task itself does not improve the box AP. Although all tricks contribute to the improvements, we decide to not use larger basis resolution because it slows down the model by 10ms per image.

We also implement the protonet module in YOLACT [3] for comparison. We include a P3 version and an FPN version. The P3 version is identical to the one used in YOLACT. For the FPN version, we first change the channel width of P3, P4, and P5 to 128 with a 3×3 convolution. Then upsample all features to s/8 and sum them up. Follow-

K	AP	AP_{50}	AP_{75}
1	30.6	52.9	31.6
2	31.2	53.4	32.3
4	**33.1**	**54.1**	**34.9**
8	33.0	53.9	**34.9**

Table 3: Number of bases: Performances of `56_K_7` models. For the configuration of one basis, we use sigmoid activation for both top and bottom features. Our model works with a small number of bases.

Features	M	Time (ms)	AP	AP_{50}	AP_{75}
C3, C5	7	74.1	33.1	54.1	34.9
	14	77.7	33.3	54.1	35.3
P3, P5	7	**72.5**	33.3	54.2	35.3
	14	76.4	**33.4**	**54.3**	**35.5**

Table 4: Bottom feature locations: Performance with bottom resolution 56×56, 4 bases and bilinear bottom interpolation. C3, C5 uses features from backbone. P3, P5 uses features from FPN.

Interpolation	M	AP	AP_{50}	AP_{75}
Nearest	7	33.3	54.2	35.3
	14	33.4	54.3	35.5
Bilinear	7	33.5	54.3	**35.7**
	14	**33.6**	**54.6**	35.6

Table 5: Top interpolation: Performance with bottom resolution 56×56, 4 bases and bilinear bottom interpolation. Nearest represents nearest-neighbour upsampling and bilinear is bilinear interpolation.

Alignment	R	M	AP	AP_{50}	AP_{75}
Nearest	28	7	30.5	53.0	31.6
	56	14	31.9	53.6	33.4
Bilinear	28	7	32.4	54.4	34.5
	56	14	**33.6**	**54.6**	**35.6**

Table 6: Bottom Alignment: Performance with 4 bases and bilinear top interpolation. Nearest represents the original RoIPool in Fast R-CNN [11] and bilinear is the RoIAlign in Mask R-CNN [12].

Bottom	Time (ms)	AP^{bb}	AP	AP_{50}	AP_{75}
DeepLabV3+	**76.5**	38.8	33.6	54.6	35.6
+semantic	**76.5**	**39.2**	34.2	54.9	36.4
+128	78.5	39.1	34.3	54.9	36.6
+s/4	86.4	**39.2**	**34.4**	**55.0**	**36.8**
Proto-P3	85.2	39.0	**34.4**	54.9	**36.8**
Proto-FPN	78.8	39.1	**34.4**	54.9	**36.8**

Table 7: Other improvements: We use `56_4_14x14` with bilinear interpolation for all models. '+semantic' is the model with semantic supervision as auxiliary loss. '+128' is the model with bottom module channel size being 256. '+s/4' means using P2,P5 as the bottom input. Decoders in DeepLab V3+ and YOLACT (Proto) are compared. 'Proto-P3' has channel width of 256 and 'Proto-FPN' of 128. Both are trained with '+semantic' setting.

Method	Backbone	Epochs	Aug.	Time (ms)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [12]	R-50	12		97.0	34.6	56.5	36.6	15.4	36.3	49.7
Mask R-CNN*		72	✓	97+	36.8	**59.2**	39.3	17.1	38.7	52.1
TensorMask [7]		72	✓	400+	35.5	57.3	37.4	16.6	37.0	49.1
BlendMask		12		**78.5**	34.3	55.4	36.6	14.9	36.4	48.9
BlendMask		36	✓	**78.5**	**37.0**	58.9	**39.7**	**17.3**	**39.4**	**52.5**
Mask R-CNN	R-101	12		118.1	36.2	58.6	38.4	16.4	38.4	52.1
Mask R-CNN*		36	✓	118+	38.3	61.2	40.8	18.2	40.6	54.1
TensorMask		72	✓	400+	37.3	59.5	39.5	17.5	39.3	51.6
SOLO [24]		72	✓	-	37.8	59.5	40.4	16.4	40.6	54.2
+deform convs [24]		72	✓	-	40.4	62.7	43.3	17.6	43.3	58.9
BlendMask		36	✓	101.8	38.4	60.7	41.3	18.2	41.5	53.3
BlendMask*		36	✓	**94.1**	39.6	61.6	42.6	22.4	42.2	51.4
+deform convs (interval = 3)		60	✓	105.0	**41.3**	**63.1**	**44.6**	**22.7**	**44.1**	**54.5**

Table 8: Quantitative results on COCO `test-dev`. We compare our BlendMask against Mask R-CNN and TensorMask. Mask R-CNN* is the modified Mask R-CNN with implementation details in TensorMask [7]. Models with 'aug.' uses multi-scale training with shorter side range $[640, 800]$. Speed for Mask R-CNN $1\times$ and BlendMask are measured with `maskrcnn_benchmark` on a single 1080Ti GPU. BlendMask* is implemented with `Detectron2`, the speed difference is caused by different measuring rules. '+deform convs (interval = 3)' uses deformable convolution in the backbone with interval 3, following [2].

Method	Backbone	NMS	Resolution	Time (ms)	AP^{bb}	AP	AP_{50}	AP_{75}
YOLACT	R-101	Fast	550×550	**34.2**	32.5	29.8	48.3	31.3
YOLACT		Fast	700×700	46.7	33.4	30.9	49.8	32.5
BlendMask-RT		Batched	$550 \times *$	47.6	**41.6**	**36.8**	**61.2**	**42.4**
Mask R-CNN	R-50	Batched	$550 \times *$	63.4	39.1	**35.3**	**56.5**	**37.6**
BlendMask-RT				**36.0**	**39.3**	35.1	55.5	37.1

Table 9: Real-time setting comparison of speed and accuracy with other state-of-the-art methods on COCO `val2017`. Metrics for YOLACT are obtained using their official code and trained model. Mask R-CNN and BlendMask models are trained and measured using `Detectron2`. Resolution $550 \times *$ means using shorter side 550 in inference. Our fast version of BlendMask significantly outperforms YOLACT in accuracy with *on par* execution time.

ing are the same as P2 version except that we reduce convolution layers by one. Auxiliary semantic loss is applied to both versions. As shown in Table 7, changing the bottom module from DeepLabv3+ to protonet does not modify the speed and performance significantly.

4.2. Main result

Quantitative results We compare BlendMask with Mask R-CNN [12] and TensorMask [7] on the COCO `test-dev` dataset[3]. We use `56_4_14` with bilinear top interpolation, the DeepLabV3+ decoder with channel width 256 and P3, P5 input. Since our ablation models are heavily under-fitted, we increase the training iterations to 270K ($3\times$ schedule), tuning learning rate down at 180K and 240K. Following Chen *et al.*'s strategy [7], we use multi-scale training with shorter side randomly sampled from $[640, 800]$. As shown in Table 8, our BlendMask outperforms both the modified Mask R-CNN with deeper FPN and TensorMask using only half of their training iterations.

[3]To make fair comparison with TensorMask, the code base that we use for main result is `maskrcnn_benchmark`. Recently released `Detectron2` fixed several issues of `maskrcnn_benchmark` (`ROIAlign` and `paste_mask`) in the previous repository and the performance is further improved.

BlendMask is also more efficient. Measured on a V100 GPU, the best R-101 BlendMask runs at 0.07s/im, *vs.* TensorMask's 0.38s/im, *vs.* Mask R-CNN's 0.09s/im [7]. Furthermore, a typical running time of our blender module is merely 0.6ms, which makes the additional time for complex scenes nearly negligible. On the contrary, for two-stage Mask R-CNN, the inference time increases by a lot if the number of predicted instances grows.

Real-time setting We design a compact version of our model, BlendMask-RT, to compare with YOLACT [3], a real-time instance segmentation method: i) the number of convolution layers in the prediction head is reduced to three, ii) and we merge the classification tower and box tower into one by sharing their features. We use Proto-FPN with four convolution layers with width 128 as the bottom module. The top FPN output P7 is removed because it has little effect on the detecting smaller objects. We train both BlendMask-RT and Mask R-CNN with the $\times 3$ schedule, with shorter side randomly sampled from $[440, 550]$.

There are still two differences in the implementation comparing to YOLACT. YOLACT resizes all images to square, changing the aspect ratios of inputs. Also, a paralleled NMS algorithm called Fast NMS is used in YOLACT.

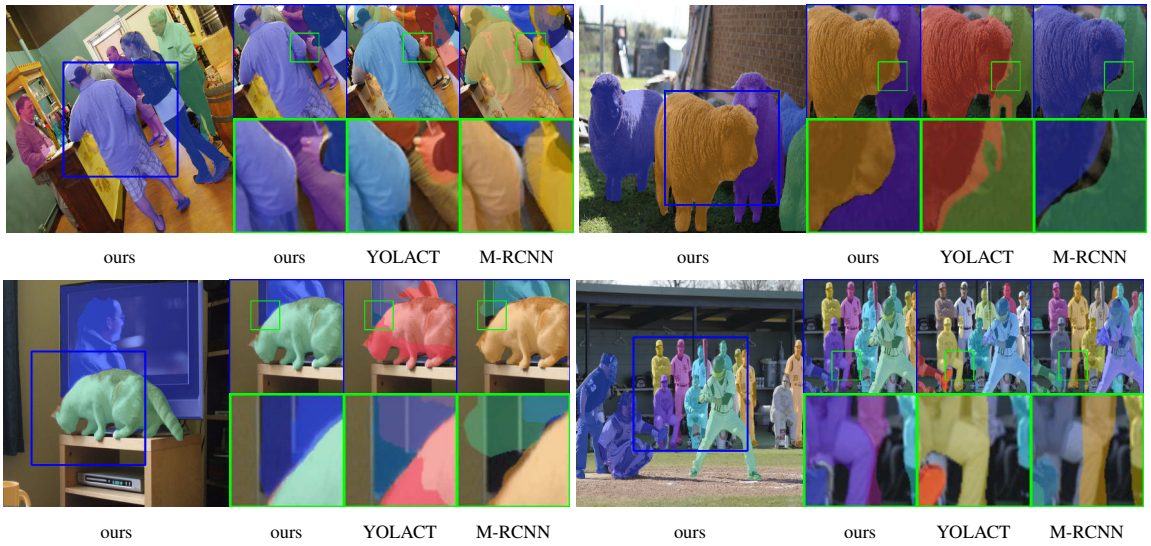


Figure 4: Detailed comparison with other methods. The large image on the left side is the segmentation result of our method. We further zoom in our result and compare against YOLACT [3] (31.2% mAP) and Mask R-CNN [12] (36.1% mAP) on the right side. Our masks are overall of higher quality.

We do not adopt these two configurations because they are not conventionally used in instance segmentation researches. In YOLACT, a speedup of 12ms is reported by using Fast NMS. We instead use the Batched NMS in `Detectron2`, which could be slower than Fast NMS but does not sacrifice the accuracy. Results in Table 9 shows that *BlendMask-RT is 7ms faster and 3.3 AP higher than YOLACT-700*. Making our model also competitive under the real-time settings.

Qualitative results We compare our model with the best available official YOLACT and Mask R-CNN models with ResNet-101 backbone. Masks are illustrated in Figure 4. Our model yields higher quality masks than Mask R-CNN. The first reason is that we predicts 56×56 masks while Mask R-CNN uses 28×28 masks. Also our segmentation module mostly utilizes high resolution features that preserve the original aspect-ratio, where Mask R-CNN also uses 28×28 features.

Note that YOLACT has difficulties discriminating instances of the same class close to each other. BlendMask can avoid this typical leakage. This is because its top module provides more detailed instance-level information, guiding the bases to capture position-sensitive information and suppressing the outside regions.

4.3. Discussions

Comparison with Mask R-CNN Similar to Mask R-CNN, we use RoIPooler to locate instances and extract features. We reduce the running time by moving the computation of R-CNN heads before the RoI sampling to generate position-sensitive feature maps. Repeated mask representation and computation for overlapping proposals are avoided. We further simplify the global map representation by replacing the hard alignment in R-FCN [9] and FCIS [16] with our attention guided blender, which needs ten times less channels for the same resolution.

Another advantage of BlendMask is that it can produce higher quality masks, since our output resolution is not restricted by the top-level sampling. Increasing the RoIPooler resolution of Mask R-CNN will introduce the following problem. The head computation increases quadratically with respect to the RoI size. Larger RoIs requires deeper head structures. Different from dense pixel predictions, RoI foreground predictor has to be aware of whole instance-level information to distinguish foreground from other overlapping instances. Thus, the larger the feature sizes are, the deeper sub-networks is needed.

Furthermore, the inference time of Mask R-CNN is strongly correlates the number of detections. In contrast, our blender module is very efficient (0.6ms on 1080 Ti). The additional inference time required after increasing the number of detections can be neglected which is suitable for real-time scenarios which require stable prediction time.

Conclusion We have devised a novel blender module for instance-level dense prediction tasks which uses both high-level instance and low-level semantic information. It is efficient and easy to integrate with different main-stream detection networks. We believe that our BlendMask is capable of serving as an alternative to Mask R-CNN [12] for many other instance-level recognition tasks such as keypoint detection.

Acknowledgements The authors would like to thank Huawei Technologies for the donation of GPU cloud computing resources.

References

[1] Anurag Arnab and Philip H. S. Torr. Bottom-up instance segmentation using deep higher-order CRFs. In *Proc. British Conf. Machine Vis.*, 2016.

[2] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT++: Better real-time instance segmentation. *arXiv preprint arXiv:1912.06218*, 2019.

[3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: real-time instance segmentation. *Proc. Int. Conf. Comp. Vis.*, abs/1904.02689, 2019.

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv Comput. Res. Repository*, abs/1708.02551, 2017.

[5] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4013–4022, 2018.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comp. Vis.*, pages 833–851, 2018.

[7] Xinlei Chen, Ross B. Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. *Proc. Int. Conf. Comp. Vis.*, abs/1903.12174, 2019.

[8] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 534–549, 2016.

[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 379–387, 2016.

[10] Cheng-Yang Fu, Tamara L. Berg, and Alexander C. Berg. IMP: instance mask projection for high accuracy semantic segmentation of things. *arXiv Comput. Res. Repository*, abs/1906.06597, 2019.

[11] Ross B. Girshick. Fast R-CNN. In *Proc. Int. Conf. Comp. Vis.*, pages 1440–1448, 2015.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. Int. Conf. Comp. Vis.*, pages 2980–2988, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 630–645, 2016.

[15] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6399–6408, 2019.

[16] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4438–4446, 2017.

[17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. Int. Conf. Comp. Vis.*, pages 2999–3007, 2017.

[18] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, pages 740–755, 2014.

[19] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity derivation and graph merge for instance segmentation. In *Proc. Eur. Conf. Comp. Vis.*, pages 708–724, 2018.

[20] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8837–8845, 2019.

[21] Pedro H. O. Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1990–1998, 2015.

[22] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 91–99, 2015.

[23] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. *Proc. Int. Conf. Comp. Vis.*, abs/1904.01355, 2019.

[24] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. *arXiv preprint arXiv:1912.04488*, 2019.