EmbedMask: Embedding Coupling for Instance Segmentation

Hui Ying[1] , **Zhaojin Huang**[2] , **Shu Liu**[3] , **Tianjia Shao**[1*] and **Kun Zhou**[1]
[1]State Key Lab of CAD&CG, Zhejiang University
[2]KuaiShou
[3]SmartMore
huiying@zju.edu.cn, {zjhuang223, liushuhust}@gmail.com, tjshao@zju.edu.cn, kunzhou@acm.org

Abstract

Current instance segmentation methods can be categorized into segmentation-based methods and proposal-based methods. The former performs segmentation first and then does clustering, while the latter detects objects first and then predicts the mask for each object proposal. In this work, we propose a single-stage method, named EmbedMask, that unifies both methods by taking their advantages, so it can achieve good performance in instance segmentation and produce high-resolution masks in a high speed. EmbedMask introduces two newly defined embeddings for mask prediction, which are *pixel embedding* and *proposal embedding*. During training, we enforce the pixel embedding to be close to its coupled proposal embedding if they belong to the same instance. During inference, pixels are assigned to the mask of the proposal if their embeddings are similar. This mechanism brings several benefits. First, the pixel-level clustering enables EmbedMask to generate high-resolution masks and avoids the complicated two-stage mask prediction. Second, the existence of proposal embedding simplifies and strengthens the clustering procedure, so our method can achieve high speed and better performance than segmentation-based methods. Without any bell or whistle, EmbedMask outperforms the state-of-the-art instance segmentation method Mask R-CNN on the challenging COCO dataset, obtaining more detailed masks at a higher speed.

1 Introduction

Instance segmentation is a fundamental and important task in computer vision. It requires finding all the objects with their categories and masks in an image. In some sense, instance segmentation can also be regarded as the combination of object detection and semantic segmentation. Hence it is a really challenging task.

Based on the top-down idea, proposal-based methods solve the object detection first, and then the segmentation task is processed based on the detected locations. As a representative, Mask R-CNN [He *et al.*, 2017] achieves outstanding results on many benchmarks to be the most popular method for instance segmentation. As a two-stage method, it uses the repooling step to extract the area of interest for the proposed objects. However, this operation results in the loss of features and the distortion of aspect ratios, so the masks it produces may not preserve fine details. Different from the proposal-based methods, segmentation-based methods treat the instance segmentation in a bottom-up way. Specifically, these methods predict the embedding for each pixel and then the clustering process is applied so that pixels with similar embeddings are grouped to form objects. Since these procedures are all done at the pixel-level directly, they do not suffer from the repooling operation. However, the bottleneck of such methods is their clustering procedure. That is, the number of clusters and the positions of cluster centers are quite difficult to be determined in these methods.

Therefore, in this work, we propose a novel instance seg-

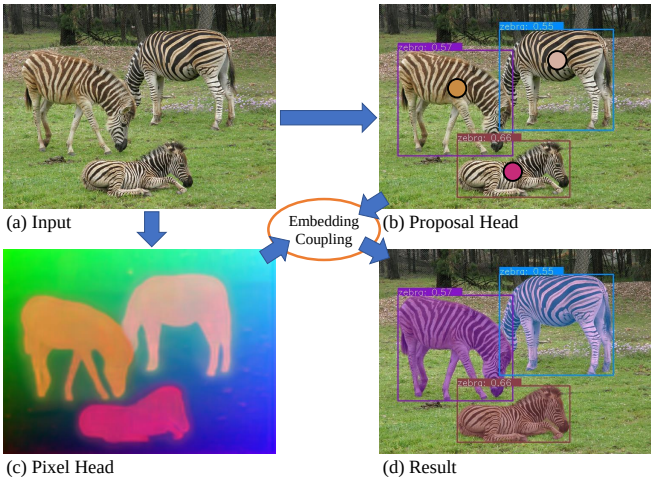

Figure 1: The pipeline of EmbedMask. (a) is an input image. (b) is the output object proposals extracted from the Proposal Head, attached with the parameters of bounding boxes, class scores, and proposal embeddings (colored dots). (c) is the output pixel embedding map extracted from Pixel Head. (d) is the final result conducted from (b) and (c) with embedding coupling. The embeddings in (b) and (c) are encoded with different colors using PCA.

*Corresponding author

mentation method, named EmbedMask. It preserves strong detection capability as the proposal-based methods, and meanwhile keeps the details of images as the segmentation-based methods. As illustrated in Fig. 1, in EmbedMask, we divide the instance segmentation pipeline into two parallel branches. "Proposal Head" follows the framework of object detection to predict object proposals, while "Pixel Head" is used to generate a pixel-level map for mask generation. To predict the mask for each object proposal, we introduce two newly defined embeddings: (1) embedding for object proposals, referred to as *proposal embedding*, which is produced by "Proposal Head"; and (2) embedding for pixels, referred to as *pixel embedding*, which is the output of "Pixel Head". A pixel embedding and a proposal embedding are considered coupled if the pixel falls inside the mask of the object proposal. During training, a pixel embedding is trained to get close to its coupled proposal embedding, and get away from other proposal embeddings. During inference, the pixel embeddings close to a proposal embedding will be clustered together to generate the mask of the object proposal. With this process, we not only avoid the difficulties in determining the cluster centers and their number but also remove the requirement of the repooling operation. To achieve better performance in embedding coupling, we explicitly divide the embedding for proposals or pixels into two parts, a spatial-aware embedding and a spatial-free embedding. The spatial-aware embedding provides spatial information, while the spatial-free embedding contains essential complementary context features. We show such factorization effectively improves the quality of mask generation. Furthermore, we predict a proposal-adaptive parameter for the object proposal to produce a certain margin for the clustering procedure. Such adaptive margins make it more suitable to conduct instance segmentation for multi-scale objects.

While being simple but effective, EmbedMask achieves superior results over Mask R-CNN, with the mask AP (Average Precision) of 38.3 vs. 38.1 in the challenging COCO dataset [Lin *et al.*, 2014] and speed of 11.7 fps vs. 6.3 fps (on an NVIDIA GeForce 2080 Ti GPU), both using the ResNet-101 [He *et al.*, 2016] as backbone networks and the same training settings. In summary, the main contributions of our work are three folds:

1. We propose a framework that unites the proposal-based and segmentation-based methods, by introducing the concepts of proposal embedding and pixel embedding.
2. Spatial-aware and spatial-free embeddings are proposed in our method to improve the quality of mask generation.
3. As a one-stage instance segmentation method, our method outperforms the state-of-the-art two-stage method Mask R-CNN in the COCO benchmark, and meanwhile, it runs at a higher speed and provides masks with a higher resolution than Mask R-CNN.

2 Related Work

Instance segmentation is a fundamental yet challenging task, which aims to predict a pixel-level mask with a category label for each instance of interest in an image. With the fast development of deep learning, a variety of methods have been proposed to solve this problem.

2.1 Proposal-based Methods

Proposal-based methods start from predicting a set of object proposals and a segmentation mask is extracted for each of these object proposals.

One of the most popular implementations for proposal-based methods is to split the instance segmentation task into two consecutive stages. Before the rise of the unified framework, [Pinheiro *et al.*, 2015] proposed DeepMask, which utilizes sliding windows to generate proposal regions and then learns to classify and segment them. Based on Faster R-CNN [Ren *et al.*, 2015], Mask R-CNN [He *et al.*, 2017] unites the tasks of region proposing and segmentation using repooling, making it the representative of two-stage instance segmentation methods. On the basis of Mask R-CNN, PANet [Liu *et al.*, 2018b] enhances the performance by merging multi-level information. MS R-CNN [Huang *et al.*, 2019] simply redefines the grading standard of instance mask. With the detection models built on top of Feature Pyramid Networks (FPN) [Lin *et al.*, 2017] as the baseline, recent two-stage instance segmentation methods achieve state-of-the-art performance. However, there still remain problems, such as the low speed and detail-missing in the masks of large objects due to the complicated network architectures and the repooling step.

Different from the above two-stage methods that employ repooling for mask prediction, other proposal-based methods provide more ideas about mask generation. To avoid re-extracting features for object proposals, [Dai *et al.*, 2016] and [Li *et al.*, 2017] generate position-sensitive mask maps for the image, and the final mask for each object proposal is fetched by assembling the maps. TensorMask [Chen *et al.*, 2019] regards the instance segmentation task as a similar problem to object detection, wherein it replaces the 3D tensors for representing the bounding boxes with the 4D tensors to represent the masks over the 2D spatial domain. Recently, CondInst [Tian *et al.*, 2020] and SOLOv2 [Wang *et al.*, 2020a] make use of conditional convolution to generate masks conditioned on the instance. CenterMask [Wang *et al.*, 2020b] predicts the local shape which is a rough mask for each instance, and then multiplies it with the global saliency map which is a detailed foreground mask map to fetch the instance-specific mask. YOLACT [Bolya *et al.*, 2019] and BlendMask [Chen *et al.*, 2020] are similar in that they predict mask bases first, and then linearly combine them to produce masks. Different from the above ones, our method utilizes embedding coupling for mask generation, which is simpler and eliminates the need of box cropping or resizing operation which is required in YOLACT, CenterMask and BlendMask. Though simple, our method achieves higher scores and higher speeds than most of the state-of-the-art methods.

2.2 Segmentation-based Methods

The segmentation-based methods consider the task of instance segmentation from another view. Like semantic segmentation, they do pixel-level predictions only. Specifically,

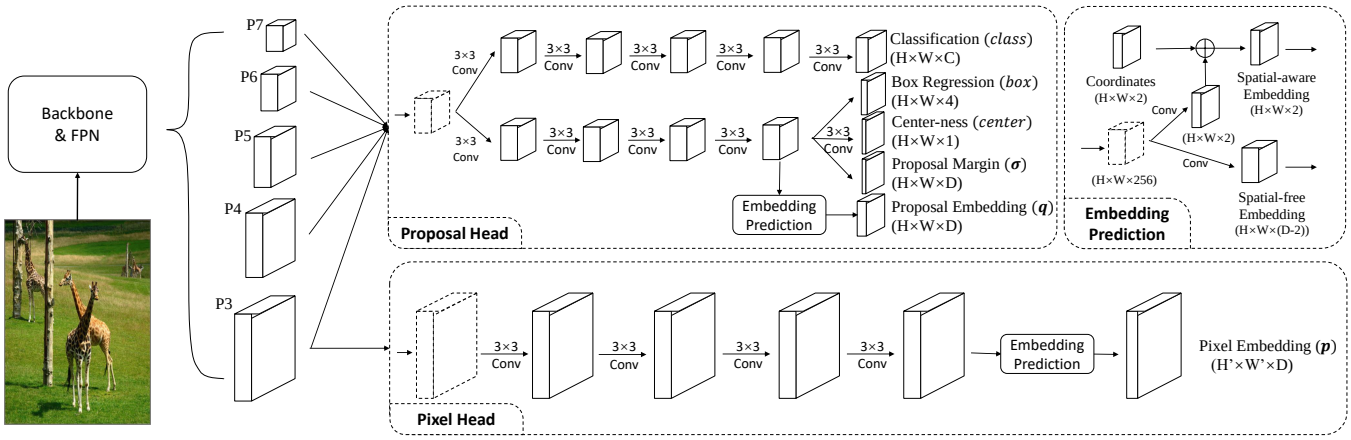

Figure 2: The detailed network architecture of EmbedMask. EmbedMask uses common backbone networks, e.g., ResNet and FPN to extract feature maps in different scales. All extracted feature maps are passed through "Proposal Head" for the prediction of proposal attributions, and the parameters of the head are shared among these feature maps. The feature map with the max size is passed through "Pixel Head". The "Embedding Prediction" module is used to predict embeddings for proposals and pixels, but its parameters are not shared between "Proposal Head" and "Pixel Head". The spatial-aware embedding and spatial-free embedding in "Embedding Prediction" will be concatenated to form the output embedding. Before the output layer, the "3×3 $conv$" in"Proposal Head" and "Pixel Head" is followed with group normalization and ReLU.

their segmentation module is used to predict the embedding for each pixel, and then clustering is applied to group the pixels for generating object masks. For separating pixels of different objects and clustering pixels of the same objects, [De Brabandere *et al.*, 2017] utilizes the discriminative loss while [Neven *et al.*, 2019] introduces a new loss function that learns extra margins for different objects. Nevertheless, during inference, these methods suffer from difficulties in clustering. To perform clustering among the embeddings, [De Brabandere *et al.*, 2017] adopts mean-shift and [Liang *et al.*, 2017] uses spectral clustering with the predicted instance number. Meanwhile [Fathi *et al.*, 2017] and [Neven *et al.*, 2019] utilize seed generation for finding the cluster centers. Such bottom-up methods can naturally fetch high-resolution masks, but their performance in perceiving instances is worse compared to proposal-based methods. Our method also uses embedding for mask generation. However, the coupling of proposal embedding and pixel embedding enables our method to perform efficient pixel clustering and achieve better performance with high-resolution masks.

3 EmbedMask

3.1 Overview

Different from original proposal-based and segmentation-based methods, our method applies a novel instance segmentation framework, which is composed of two parallel modules, 'Proposal Head" for object-level detection and 'Pixel Head" for pixel-level embedding prediction. "Proposal Head" serves to extract attributes for object proposals (e.g. object class and bounding box), and it predicts embedding for each proposal additionally. Meanwhile, "Pixel Head" aims to produce a pixel-level embedding map. Then, the proposal embedding and pixel embedding can be combined to generate masks. In particular, during inference, if a pixel embedding is close enough to the proposal embedding, they are considered to be coupled. After the embedding coupling is applied to every pixel and proposal, the masks of object proposals are generated.

In contrast to the previous segmentation-based methods which do instance segmentation by clustering pixels with similar embeddings, our method utilizes the coupling of proposal embedding and pixel embedding, which can effectively get rid of the difficulties of clustering, such as finding the locations and number of cluster centers.

3.2 Network Architecture

In practice, we use the state-of-the-art object detection method FCOS [Tian *et al.*, 2019b] as our detection baseline, which is the most recent one-stage object detection method. Please note that our method can also deploy other detection frameworks.

Figure 2 shows the network architecture of EmbedMask. After the input image is passed through the ResNet [He *et al.*, 2016] and FPN (Feature Pyramid Networks) [Lin *et al.*, 2017], the generated features are fed into two different network branches, i.e., "Proposal Head" and "Pixel Head". The "Proposal Head" takes the feature map from each level of FPN as input, and the features are passed through two subbranches with the same hidden layer architecture, which consists of four 3×3 $conv$ layers. Five final feature maps are extracted as the output. Among the five feature maps, three of these are introduced in FCOS [Tian *et al.*, 2019b], i.e., classification, center-ness, and box regression. The other two feature maps are the new components introduced in our work, i.e., proposal embedding and proposal margin, referred to as $\boldsymbol{q}$ and $\boldsymbol{\sigma}$. These five feature maps are united to represent the attributions of object proposals. Specifically, the values at the same location j of these feature maps can be grouped as a tuple $\{class_j, box_j, center_j, \boldsymbol{q}_j, \boldsymbol{\sigma}_j\}$ that represents the

parameters of one proposal. In parallel with the "Proposal Head", "Pixel Head" takes as input the largest feature map of FPN, i.e., P3, and also consists of four 3×3 $conv$ hidden layers and an output layer. The output is the pixel embedding map, referred to as $\boldsymbol{p}$, whose size is one-eighth of the size of input image. For each location j in the map, its value $\boldsymbol{p}_j$ represents the embedding for the pixel.

3.3 Embedding Coupling

To measure the similarity between proposal embedding and pixel embedding, we need a function to convert the embedding distance to the probability that the pixel belongs to the proposal. Inspired by [Neven *et al.*, 2019], we adopt the RBF (Radial Basis Function) kernel function:

$$\phi(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{\sigma}) = \exp\left(-\sum_{i}^{D} \frac{(\boldsymbol{p}_i - \boldsymbol{q}_i)^2}{2\boldsymbol{\sigma}_i^2}\right). \tag{1}$$

Here $\boldsymbol{q}$ is the proposal embedding for one proposal and $\boldsymbol{p}$ is the pixel embedding for one pixel, and $\boldsymbol{\sigma}$ is a standard variance. All of them are vectors with dimension D. The function measures the probability based on the L2 distance between proposal embedding and pixel embedding. When $\boldsymbol{p}$ is close to $\boldsymbol{q}$, the probability is close to 1, otherwise 0. The $\boldsymbol{\sigma}$ controls the shape of the similarity matrix, i.e., to reach a high similarity a larger $\boldsymbol{\sigma}$ can tolerate a larger distance between $\boldsymbol{p}$ and $\boldsymbol{q}$. In practice, we predict the $\boldsymbol{\sigma}$ for each object proposal like the proposal embedding, and we name it as *proposal margin* since it determines how close the two embeddings are to treat the proposal and pixel as coupled in inference. Such flexible proposal margins are necessary as the pixel embeddings coupled with different proposal embeddings may have different distributions.

3.4 Spatial-aware Embedding

In EmbedMask, for each proposal embedding $\boldsymbol{q}$ or pixel embedding $\boldsymbol{p}$, there are two components, i.e., spatial-aware embedding $(\boldsymbol{q}_a, \boldsymbol{p}_a)$ and spatial-free embedding $(\boldsymbol{q}_f, \boldsymbol{p}_f)$.

As illustrated in the top right block of Fig. 2, the spatial-free embedding ($\boldsymbol{q}_f$ or $\boldsymbol{p}_f$) with dimension = $D-2$ is extracted from the fully convolutional networks directly. It is spatial-free because of the spatial-invariant nature of fully convolutional networks. The spatial-aware embedding fuses the coordinate information with the addition operation. For example, for one pixel embedding $\hat{\boldsymbol{p}_a}$ (dimension = 2) which is directly predicted from the fully convolutional networks, the spatial-aware pixel embedding $\boldsymbol{p}_a$ is obtained via:

$$\boldsymbol{p}_a = \hat{\boldsymbol{p}_a} + \boldsymbol{u}. \tag{2}$$

where $\boldsymbol{u}$ is the location of the pixel embedding in the original image. This operation is the same for the spatial-aware proposal embedding $\boldsymbol{q}_a$. To compute the probability for embedding coupling, we also split the proposal margin into $\boldsymbol{\sigma}_a$ (dimension=2) and $\boldsymbol{\sigma}_f$ (dimension=$D-2$).

With the spatial-aware and spatial-free pixel embeddings $\boldsymbol{p}_a, \boldsymbol{p}_f$ and proposal embeddings $\boldsymbol{q}_a, \boldsymbol{q}_f$, as well as the proposal margins $\boldsymbol{\sigma}_a, \boldsymbol{\sigma}_f$, the final probability that a pixel belongs to a proposal is:

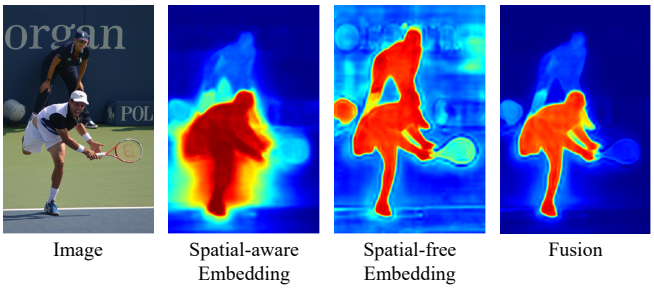

Figure 3: Mask probability map from spatial-aware embedding, spatial-free embedding and the fusion.

$$P = \phi(\boldsymbol{p}_a, \boldsymbol{q}_a, \boldsymbol{\sigma}_a) \cdot \phi\big(\boldsymbol{p}_f, \boldsymbol{q}_f, \boldsymbol{\sigma}_f\big). \tag{3}$$

which is the product of the RBF kernels from spatial-aware embedding and spatial-free embedding. As shown in Figure 3, the spatial-aware embedding gives a coarse mask in the spatial space, and the spatial-free embedding ignores the spatial information but gives a more detailed mask. The fusion of them gives a mask that is closer to the ground-truth.

3.5 Training and Inference

Equation 3 gives the computation of the probability that one pixel belongs to one proposal. Consequently, given an instance, when the probability equation is applied to all pixels in the pixel embedding map, a foreground/background probability map for the instance is produced. During training, this probability map can be optimized with a binary classification loss by comparing it with the ground-truth mask map of the instance:

$$L_{mask} = \mathcal{L}\left(\mathcal{P}_k, \mathcal{M}_k\right). \tag{4}$$

where k is the index of each instance. $\mathcal{P}_k$ represents the computed foreground/background probability map for the instance k, and $\mathcal{M}_k$ represents the corresponding ground-truth mask map. $\mathcal{L}(\cdot)$ is the binary classification loss function. In practice we use the Lovász-hinge loss [Yu and Blaschko, 2015] for better performance. The remaining question now is which proposal is responsible for generating $\mathcal{P}_k$ and to which ground-truth instance the proposal corresponds. As introduced in Section 3.2, a tuple $\{class_j, box_j, center_j, \boldsymbol{q}_j, \boldsymbol{\sigma}_j\}$ is produced for each location j on the predicted feature maps from "Proposal Head". To produce $\mathcal{P}_k$, we use the $\boldsymbol{q}_j$ and $\boldsymbol{\sigma}_j$ whose corresponding box_j is positive in the training of FCOS, and the corresponding $\mathcal{M}_k$ is just from the matched ground-truth instance. For the alignment of $\mathcal{P}_k$ and $\mathcal{M}_k$ when computing the mask loss during training, we resize the predicted mask probability map and the ground-truth mask to be a quarter of the input image in length, using bilinear interpolation. We do the same for the mask probability map and then re-scale it to the initial size to obtain the mask during inference.

EmbedMask is optimized in an end-to-end way using a multi-task loss. Apart from the original classification loss

method	backbone	ms	rc	epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP^{box}
Mask R-CNN*	R-50-FPN			12	34.6	56.5	36.6	15.3	36.3	49.7	38.0
Mask R-CNN*	R-101-FPN	✓		36	38.1	60.9	40.7	18.4	40.2	53.4	42.6
PANet	R-50-FPN	✓		22	38.2	60.2	41.4	19.1	41.1	52.6	-
MS R-CNN	R-101-FPN			18	**38.3**	58.8	41.5	17.8	40.4	54.4	-
TensorMask	R-101-FPN	✓		72	37.3	59.5	39.5	17.5	39.3	51.6	41.6
YOLACT-700	R-101-FPN	✓	✓	48	31.2	50.6	32.8	12.1	33.3	47.1	-
PolarMask	R-101-FPN	✓		24	32.1	53.7	33.1	14.7	33.8	45.3	-
CenterMask	R-101-FPN			24	36.1	58.7	38.0	16.5	38.4	51.2	-
EmbedMask	R-50-FPN			12	34.8	55.1	37.4	15.5	37.4	49.4	38.2
EmbedMask	R-101-FPN	✓		36	**38.3**	59.3	41.2	18.1	40.8	53.9	42.3

Table 1: Quantitative comparison with the state-of-the-art methods. 'ms' and 'rc' mean multi-scale augmentation and random cropping augmentation for training. For a fair comparison, Mask R-CNN* and EmbedMask are both implemented with the maskrcnn-benchmark.

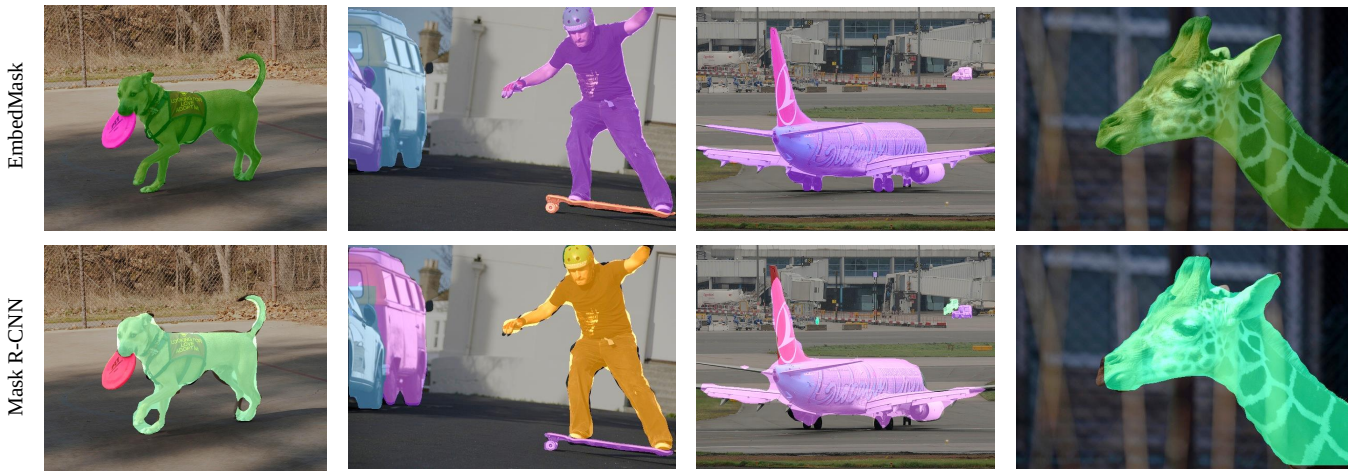

Figure 4: Qualitative comparison with Mask R-CNN.

L_{cls}, center-ness loss L_{center} and box regression loss L_{box} in FCOS, the additional loss L_{mask} is introduced for mask prediction. They are jointly optimized by

$$L = L_{cls} + L_{box} + L_{center} + \lambda_1 L_{mask}. \quad (5)$$

Here λ_1 is a loss weight for mask loss, and we set $\lambda_1 = 0.5$ by default.

During inference, the embedding coupling procedure is more clear. Given an input image, it will go through the networks to extract the proposal attributions from "Proposal Head" and the pixel embedding map from "Pixel Head". After NMS (Non-Maximum Suppression) is applied to the proposal attributions, each surviving proposal is attached with a bounding box, a category with a related score, a proposal embedding $\boldsymbol{q}$, and a proposal margin $\boldsymbol{\sigma}$. For each pixel in the pixel embedding map, the probability that the pixel belongs to a proposal is computed as in Equation (3). This probability is then converted to a binary value using a fixed threshold. In this way, the final mask for each object proposal is produced. The selection of the threshold is discussed in the ablation study.

4 Experiments

4.1 Experimental Settings

We follow the settings of FCOS [Tian *et al.*, 2019b] in our experiments, which chooses the large-scale detection benchmark COCO, and uses the COCO *trainval35k* split (115K images) for training, *minival* split (5K images) for ablation study and *test-dev* (20K images) for reporting the main results. Unless noted, the input images are resized with the shorter side being 800 while the longer side being no longer than 1333. Nevertheless, in Table 1 and Table 2, multi-scale augmentation is applied for fair comparison and better results. Multi-scale augmentation makes the shorter side of input images range in [640,800] while the longer side less than or equal to 1333.

We train all the models with SGD using an initial learning rate of 0.01 and batch size of 16, with constant warm-up of 500 iterations. ResNet-50 [He *et al.*, 2016] is used as our backbone network for ablation study, and ResNet-101 is used for comparison with state-of-the-art methods. The backbone networks are initialized with the pre-trained ImageNet [Deng *et al.*, 2009] weights. The models are trained for 12 epochs ($90k$ iterations) by default, but more epochs are applied when

method	backbone	epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP^{box}	$time(ms)$
CondInst	R-50-FPN	12	**36.4**	57.6	38.8	18.9	38.8	48.7	40.3	57.2
BlendMask	R-50-FPN	12	36.0	56.9	38.4	19.4	38.5	47.1	40.5	59.2
SOLOv2	R-50-FPN	12	35.6	56.0	38.0	7.5	55.3	70.0	-	<u>56.8</u>
EmbedMask	R-50-FPN	12	<u>36.3</u>	57.0	39.0	18.3	39.0	49.0	40.1	**56.4**
CondInst	R-101-FPN	36	**40.1**	61.9	43.0	21.7	42.8	53.1	44.7	75.6
BlendMask	R-101-FPN	36	39.6	61.4	42.6	22.1	42.3	51.1	44.7	76.2
SOLOv2	R-101-FPN	36	39.6	60.5	42.8	9.9	58.3	72.3	-	<u>73.3</u>
EmbedMask	R-101-FPN	36	<u>40.0</u>	61.6	43.0	21.1	43.1	53.6	44.6	**72.4**

Table 2: Quantitative comparison with the most recent state-of-the-art methods. All these methods are implemented based on AdelaiDet for a fair comparison.

training with the ResNet-101 backbone, as shown in Table 1. For $90k$ training iterations (12 epochs), the learning rate is reduced by a factor of 10 at iteration $60k$ and $80k$. For $270k$ training iterations (36 epochs), the learning rate is reduced by a factor of 10 at iteration $210k$ and $250k$. In the main results, we set embedding dim $D = 16$.

4.2 Main Results

Quantitative Comparison

As shown in Table 1, we compare the quantitative results of EmbedMask with the state-of-the-art methods, including one-stage methods (TensorMask [Chen *et al.*, 2019], YOLACT [Bolya *et al.*, 2019], PolarMask[Xie *et al.*, 2020], CenterMask[Wang *et al.*, 2020b]) and two-stage methods (Mask R-CNN [He *et al.*, 2017], PANet [Liu *et al.*, 2018b], MS R-CNN[Huang *et al.*, 2019]). The listed results are all trained with the ResNet-50 or ResNet-101 as the backbone for fairness. We evaluate the instance segmentation results using mask average precision (AP), AP at IoU 0.5 (AP_{50}) and 0.75 (AP_{75}), AP for objects at different sizes (AP_S, AP_M, AP_L), and box average precision (AP^{box}). We can see EmbedMask achieves the best performance among these methods. Especially, with the same training settings and under the same framework (maskrcnn-benchmark [Massa and Girshick, 2018]), EmbedMask achieves better results than Mask R-CNN, which demonstrates the efficiency of our framework. Please note our framework is a general one, and the performance can be further improved if equipped with other superior network architectures.

Table 2 shows the comparison with the most recent state-of-the-art methods (CondInst [Tian *et al.*, 2020], BlendMask [Chen *et al.*, 2020], and SOLOv2 [Wang *et al.*, 2020a]). These methods as well as ours all do instance segmentation from one complete feature map within the proposal-based framework, but different strategies are utilized to generate instance-specific masks. For a fair comparison, all the methods listed in the table are implemented based on AdelaiDet [Tian *et al.*, 2019a] with the same training settings. We further modify our network architecture so that it has the same architecture units as these methods. Specifically, the methods of CondInst, BlendMask and SOLOv2 all fuse the feature maps from different FPN levels in their "Mask Branch", so we do the same in our "Pixel Head". We also use the auxiliary semantic segmentation task to help with the mask prediction like CondInst and BlendMask. From Table 2, we can find that our method achieves better results than BlendMask and SOLOv2 and is comparable with CondInst. Specifically, our method is slightly better than CondInst at predicting masks for large objects, but slightly weaker than CondInst at small objects.

Qualitative Comparison

Fig. 4 visualizes the comparison of mask quality between Mask R-CNN and EmbedMask, and all of these results are from the models trained with multi-scale augmentation for 36 epochs. We can observe that our method can provide more detailed masks than Mask R-CNN with sharper and smoother boundaries. Also for the slim object parts (e.g., dog feet), our method can generate more accurate masks than Mask R-CNN. This is because our method does not use the repooling operation and can avoid missing details.

Speed Analysis

Based on the maskrcnn-benchmark implementation, EmbedMask runs in 15.1 fps with the backbone of ResNet-50 and 11.7 fps with the backbone of ResNet-101, using a single NVIDIA GeForce 2080 Ti GPU. Under the same condition, the speed of Mask R-CNN is 6.7 fps and 6.3 fps respectively. Hence our method can run faster than Mask R-CNN. Specifically, when using ResNet-50 as the backbone, the inference time of EmbedMask is about 66.2 ms, mainly consisting of the time for backbone networks (26.8 ms), head networks (25.3 ms), box prediction (8.4 ms), and mask prediction (2.5 ms). Compared to FCOS, the additional time cost of our method mainly comes from the forward time for "Pixel Head" (7.3 ms) and the time for mask prediction (2.5 ms), so our method brings only a little extra time overhead to the detector, making it possible to run faster than the two-stage methods. It is also worth noting that we can replace the FCOS with any other object detector and obtain a similar running performance as the detector.

The time performance comparison with the most recent state-of-the-art methods are also reported in Table 2 where a single NVIDIA GeForce 2080Ti GPU is used. We can see that our method is the fastest among these methods. In fact, with similar network architectures, the inference time of all the methods is similar. The main time difference comes from mask production. The results show that embedding coupling is an efficient method for mask prediction.

margin	D	AP
fixed	8	33.9
learnable	4	34.4
	8	34.8
	16	34.8

sa.e.	sf.e.	cc	AP
✓			30.8
	✓		33.3
✓	✓		34.8
		✓	34.4

Table 3: Left: ablation study of proposal margin and embedding dimension. Right: ablation study of embedding components and coordconv layer. 'sa.e.' means spatial-aware embedding. 'sf.e.' means spatial-free embedding. 'cc' means coordconv layer.

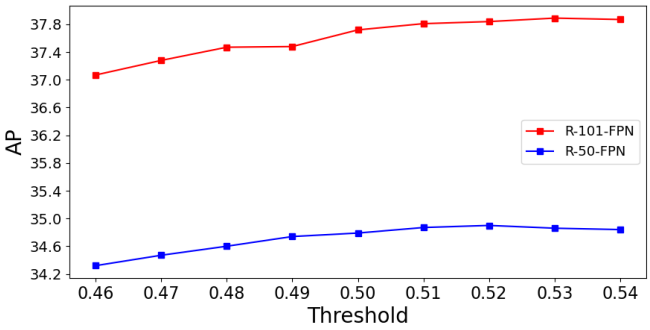

Figure 5: Mask AP with different mask thresholds.

4.3 Ablation Study

Learnable Proposal Margin. In EmbedMask, the margin is adaptive for each object proposal, which essential for good performance. To prove this, we compare the results of EmbedMask with learnable margins and with constant margins (see the supplementary material for details). Table 3 validates the adaptive margins can improve the mask generation.

Embedding Dimension. Table 3 also shows the results of EmbedMask with different embedding dimensions. When the embedding dimension is increased from 4 to 8, the mask AP increases by 0.4. When we further increase the dimension to 16, the mask AP remains similar. This shows EmbedMask is robust to the embedding dimension.

Spatial-aware Embedding. The embedding in EmbedMask consists of spatial-aware embedding and spatial-free embedding. The results in Table 3 show the effects of using either and both of the two embeddings. We can find that EmbedMask performs much better when the two embeddings are fused. We also compared with CoordConv [Liu *et al.*, 2018a] which implicitly fuses the spatial information in the convolution layer, where we replace the "Embedding Prediction" module with the CoorConv layer. The results show that our explicit fusion of spatial information performs better.

Mask Threshold. During inference, the probability map needs to be converted to a binary map using a fixed threshold. To obtain a suitable threshold, we evaluate the mask performance with different mask thresholds, as illustrated in Figure 5. We find that when the threshold is 0.52 or 0.53, the mask AP is the highest. Hence we use 0.52 for the ResNet-50 backbone and 0.53 for the ResNet-101 backbone.

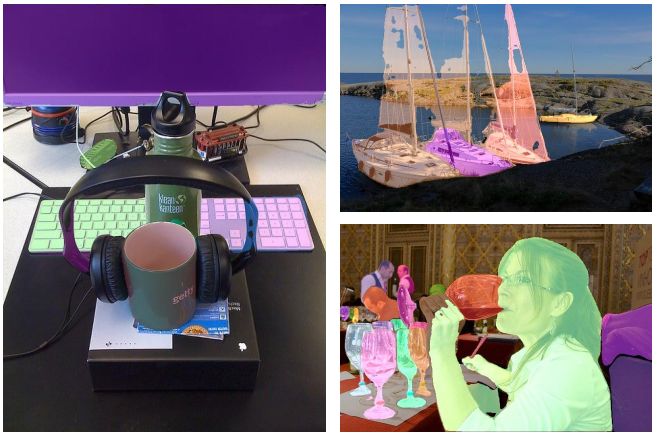

Figure 6: Failure cases.

5 Failure Cases Discussion

Figure 6 shows some of the typical failure cases in our method. In the left image, we can see that the whole keyboard is segmented into two parts. It is because the keyboard is overlapped by two bottles in the middle, and each isolated part is detected as one keyboard wrongly. The top right image shows a difficult case where some parts of the object are extremely thin. Although our method can produce sharper masks than Mask R-CNN, the mast of the boat is still a difficult case for segmentation. In the bottom right image, our method produces an incomplete mask for the hand of the front person. We observe there are multiple instance masks near the hand. The complicated nearby environment may cause the incorrect pixel embedding prediction and result in a wrong mask prediction.

6 Conclusion

We have proposed a single-stage instance segmentation method, named EmbedMask. It unites the advantages of proposal-based and segmentation-based methods, by introducing the novel proposal embedding and pixel embedding. As the key mechanism, embedding coupling measures the similarity between the pixel embedding and proposal embedding for mask generation, which eliminates the repooling operation and searching for cluster centers, and produces masks with well-preserved details. We further improve the results with the proposal-adaptive margins and the explicit division-and-fusion of spatial-aware and spatial-free embedding. Being simple but effective, EmbedMask achieves better performance than the two-stage methods and runs at a higher speed.

Acknowledgments

We thank anonymous reviewers for their valuable comments. The work was supported by NSF China (No. 61772462, No. U1736217) and the 100 Talents Program of Zhejiang University.

References

[Bolya *et al.*, 2019] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *ICCV 2019*, pages 9157–9166, 2019.

[Chen *et al.*, 2019] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *ICCV 2019*, pages 9157–9166, 2019.

[Chen *et al.*, 2020] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *CVPR*, pages 8573–8581, 2020.

[Dai *et al.*, 2016] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *ECCV 2016*, pages 3150–3158, 2016.

[De Brabandere *et al.*, 2017] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR 2009*, pages 248–255, 2009.

[Fathi *et al.*, 2017] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR 2016*, pages 770–778, 2016.

[He *et al.*, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV 2017*, pages 2961–2969, 2017.

[Huang *et al.*, 2019] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR 2019*, pages 6409–6418, 2019.

[Li *et al.*, 2017] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR 2017*, pages 2359–2367, 2017.

[Liang *et al.*, 2017] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *TPAMI*, 40(12):2978–2991, 2017.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV 2014*, pages 740–755, 2014.

[Lin *et al.*, 2017] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR 2017*, pages 2117–2125, 2017.

[Liu *et al.*, 2018a] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NIPS 2018*, pages 9605–9616, 2018.

[Liu *et al.*, 2018b] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *ICCV 2018*, pages 8759–8768, 2018.

[Massa and Girshick, 2018] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: October 16, 2019.

[Neven *et al.*, 2019] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR 2019*, pages 8837–8845, 2019.

[Pinheiro *et al.*, 2015] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *NIPS 2015*, pages 1990–1998, 2015.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS 2015*, pages 91–99, 2015.

[Tian *et al.*, 2019a] Zhi Tian, Hao Chen, Xinlong Wang, Yuliang Liu, and Chunhua Shen. AdelaiDet: A toolbox for instance-level recognition tasks. https://git.io/adelaidet, 2019. Accessed: May 1, 2021.

[Tian *et al.*, 2019b] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV 2019*, pages 9627–9636, 2019.

[Tian *et al.*, 2020] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. *arXiv preprint arXiv:2003.05664*, 2020.

[Wang *et al.*, 2020a] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *NIPS*, 33, 2020.

[Wang *et al.*, 2020b] Yuqing Wang, Zhaoliang Xu, Hao Shen, Baoshan Cheng, and Lirong Yang. Centermask: single shot instance segmentation with point representation. In *CVPR*, pages 9313–9321, 2020.

[Xie *et al.*, 2020] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, pages 12193–12202, 2020.

[Yu and Blaschko, 2015] Jiaqian Yu and Matthew Blaschko. Learning submodular losses with the lovász hinge. In *ICML 2015*, pages 1623–1631, 2015.