

BANet: Bidirectional Aggregation Network with Occlusion Handling for Panoptic Segmentation Supplementary Material

Yifeng Chen¹, Guangchen Lin¹, Songyuan Li¹, Bourahla Omar¹, Yiming Wu¹,
Fangfang Wang¹, Junyi Feng¹, Mingliang Xu², Xi Li^{1*}

¹Zhejiang University, ²Zhengzhou University
{yifengchen, aaronlin, leizungjyun, xilizju}@zju.edu.cn

1. Overview

The structure of the supplementary material is arranged as follows. In Section 2, we conduct a thorough comparison between operators of our RoInlay and RoUpsample. In Section 3, we give a video introduction to our model’s pipeline.

2. Comparision between RoInlay and RoUpsample

In this section, we compare three patch-recovering operators, *i.e.*, RoInlay, RoUpsample and Avg RoUpsample (a modified version of RoUpsample) from three aspects, namely visual effect, runtime and experiment results.

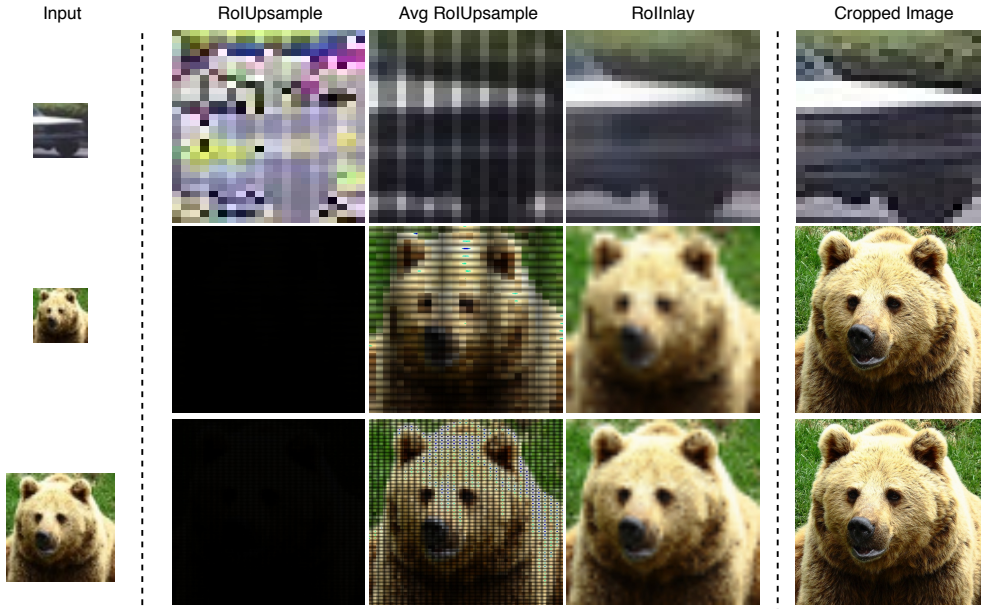


Figure 1. Comparison of RoInlay, RoUpsample and its modified version Avg RoUpsample. All of them take the output of RoIAlign as input. The output size of RoIAlign is set to 28×28 for the first and second row and 56×56 for the third row. The object in the first row is small ($< 28 \times 28$), while the one in the second and third row is large ($> 56 \times 56$).

*Corresponding author, xilizju@zju.edu.cn

Visual Effect A better patch-recovering operator has an output that looks more similar to the original cropped image. To give a visual comparison between RoIInlay and RoIUpsample, we test them on raw images (RGB). Specifically, we first use RoIAlign to obtain a resized patch of an object, and then feed it to both operators to recover its resolution. Moreover, a modified version of RoIUpsample, named as Avg RoIUpsample, is provided. It replaces the summation in RoIUpsample with averaging. As shown in Figure 1, our RoIInlay performs better no matter what object area and what output setting of RoIAlign is, while either RoIUpsample or its modified version suffers from black stripes. These black stripes are formed by sampling “holes” of RoIUpsample, as described before. The output of RoIUpsample is extremely different since it sums values from multiple reference points, leading to the change of the scale of values.

Runtime To test the speed of RoIInlay, we record its execution time and compare it with RoIUpsample’s on GTX 1080Ti. As shown in Table 1, RoIInlay is faster than RoIUpsample under various settings. The speedup ratio grows to $\times 3.60$ when object size is set to 128. In the COCO dataset, the average object size is 98×98 , indicating that we can obtain about $\times 3$ speedup with RoIInlay.

#Objects	Object size	Output Size	RoIUpsample (ms)	RoIInlay (ms)	Speed Up
50	28×28	300×300	3.65	2.17	$\times 1.68$
100	28×28	300×300	6.8	3.9	$\times 1.74$
100	28×28	800×800	9.7	6.7	$\times 1.45$
100	56×56	300×300	34.9	12.1	$\times 2.88$
100	128×128	300×300	440.5	122.4	$\times 3.60$

Table 1. Speed Comparison between RoIInlay and RoIUpsample on GTX 1080Ti. The input of both operators are tensors of 512 channels. They are resized according to object sizes and put into a tensor of output size as output.

Experimental Results We test these two operators on the model with only SIM module to show how it affects the actual performance. As shown in Table 2, RoIUpsample will hurt the segmentation quality (SQ) by about 1%.

Operator	PQ	SQ	RQ
RoIUpsample	39.4	77.0	48.7
RoIInlay	39.5	78.0	48.7

Table 2. Experimental results of RoIInlay and RoIUpsample. Both operators are applied to the model with only SIM module. The random seed and training schedule is set to the same.

3. Video Introduction

We present a video introduction, *intro.mp4*, to our approach. In this video, we depict the pipeline of our method and briefly show the procedure of each module.