CARAFE: Content-Aware ReAssembly of FEatures

Jiaqi Wang[1] Kai Chen[1] Rui Xu[1] Ziwei Liu[1] Chen Change Loy[2] Dahua Lin[1]
[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong
[2]Nanyang Technological University
{wj017,ck015,xr018,dhlin}@ie.cuhk.edu.hk zwliu.hust@gmail.com ccloy@ntu.edu.sg

Abstract

Feature upsampling is a key operation in a number of modern convolutional network architectures, e.g. *feature pyramids. Its design is critical for dense prediction tasks such as object detection and semantic/instance segmentation. In this work, we propose Content-Aware ReAssembly of FEatures (CARAFE), a universal, lightweight and highly effective operator to fulfill this goal. CARAFE has several appealing properties: (1) Large field of view. Unlike previous works (*e.g. *bilinear interpolation) that only exploit sub-pixel neighborhood, CARAFE can aggregate contextual information within a large receptive field. (2) Content-aware handling. Instead of using a fixed kernel for all samples (*e.g. *deconvolution), CARAFE enables instance-specific content-aware handling, which generates adaptive kernels on-the-fly. (3) Lightweight and fast to compute. CARAFE introduces little computational overhead and can be readily integrated into modern network architectures. We conduct comprehensive evaluations on standard benchmarks in object detection, instance/semantic segmentation and inpainting. CARAFE shows consistent and substantial gains across all the tasks (1.2%, 1.3%, 1.8%, 1.1db respectively) with negligible computational overhead. It has great potential to serve as a strong building block for future research.* [1]

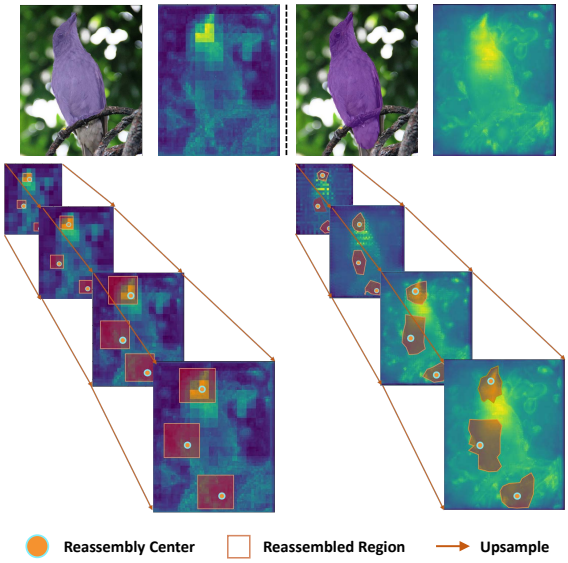

Figure 1: **Illustration of CARAFE working mechanism**. **Left:** Multi-level FPN features from Mask R-CNN (left to dotted line) and **Right:** Mask R-CNN with CARAFE (right to dotted line). For sampled locations, this figure shows the accumulated reassembled regions in the top-down pathway of FPN. Information inside such a region is reassembled into the corresponding reassembly center.

1. Introduction

Feature spatial upsampling is one of the most fundamental operations in deep neural networks. On the one hand, for the decoders in dense prediction tasks (*e.g.* super resolution [5, 14], inpainting [9, 22] and semantic segmentation [29, 3]), the high-level/low-res feature map will be upsampled to match the high resolution supervision. On the other hand, feature upsampling is also involved in fusing a high-level/low-res feature map with a low-level/high-res feature map, which is widely adopted in many state-of-the-art architectures, *e.g.* Feature Pyramid Network [15], U-Net [24] and Stacked Hourglass Networks [20]. Therefore, designing effective feature upsampling operator becomes a critical issue.

The most widely used feature upsampling operators are the nearest neighbor and bilinear interpolation, which adopt spatial distance between pixels to guide the upsampling process. However, nearest neighbor and bilinear interpolation only consider sub-pixel neighborhood, failing to capture the rich semantic information required by dense prediction tasks. Another route toward adaptive upsampling is deconvolution [21]. A deconvolution layer works as an inverse operator of a convolution layer, which learns a set of

[1]Code will be available.

instance-agnostic upsampling kernels. However, it has two major drawbacks. Firstly, a deconvolution operator applies the same kernel across the entire image, regardless of the underlying content. This restricts its capability of responding to local variations. Secondly, it comes with a large number of parameters and thus heavy computational workload when a large kernel size is used. This makes it difficult to cover a larger region that goes beyond a small neighborhood, thus limiting its expressive power and performance.

In this work, we move beyond these limitations, and seek a feature upsampling operator that is capable of 1) aggregating information within large receptive field, 2) adapting to instance-specific contents on-the-fly, and 3) maintaining computationally efficiency. To this end, we propose a lightweight yet highly effective operator, called *Content-Aware ReAssembly of FEatures (CARAFE)*. Specifically, CARAFE reassembles the features inside a predefined region centered at each location via a weighted combination, where the weights are generated in a content-aware manner. Furthermore, there are multiple groups of such upsampling weights for each location. Feature upsampling is then accomplished by regularly rearrangement the generated features as a spatial block.

Note that these spatially adaptive weights are not learned as network parameters. Instead, they are predicted on-the-fly, using a lightweight fully-convolutional module with softmax activation. Figure 1 reveals the working mechanism of CARAFE. After upsampled by CARAFE , a feature map can represent the shape of an object more accurately, so that the model can predict better instance segmentation results. Our CARAFE not only upsamples the feature map spatially, but also learns to refine it semantically.

To demonstrate the universal effectiveness of CARAFE , we conduct comprehensive evaluation across a wide range of dense prediction tasks, *i.e*., object detection, instance segmentation, semantic segmentation, image inpainting, with mainstream architectures. CARAFE can boost the performance of Faster RCNN [23] by 1.2% AP in object detection and Mask RCNN [7] by 1.3% AP in instance segmentation on MS COCO [16] test-dev 2018. CARAFE further improves UperNet [27] by 1.8% mIoU on ADE20k [32, 33] val in semantic segmentation, and improves Global&Local [9] by 1.1 db of PSNR on Places [31] val in image inpainting. When upsampling an $H \times W$ feature map with 256 channels by a factor of 2, the introduced computational overhead by CARAFE is only $H * W * 199k$ FLOPs, *vs*., $H * W * 1180k$ FLOPs of deconvolution. The substantial gains on all the tasks demonstrate that CARAFE is an effective and efficient feature upsampling operator that has great potential to serve as a strong building block for future research.

2. Related Work

Upsampling Operators. The most commonly used upsampling methods are nearest neighbor and bilinear interpolations. These interpolations leverage distances to measure the correlations between pixels, and hand-crafted upsampling kernels are used in them. In deep learning era, several methods are proposed to upsample a feature map using learnable operators. For example, deconvolution [21], which is an inverse operator of a convolution, is the most famous among those learnable upsamplers. Pixel Shuffle [25] proposes a different upsampler which reshapes depth on the channel space into width and height on the spatial space. Recently, [18] proposed guided upsampling (GUM) which performs interpolation by sampling pixels with learnable offsets. However, these methods either exploit contextual information in a small neighborhood, or require expensive computation to perform adaptive interpolation.

Dense Prediction Tasks. Object detection is the task of localizing objects with bounding-boxes and instance segmentation further requires the prediction of instance-wise masks. Faster-RCNN [23] introduces Region Proposal Network (RPN) for end-to-end training. FPN [15] can detect proposals at multiple output layers for alleviating the scale mismatch between RPN receptive fields and actual object size. By adding an extra branch into Faster R-CNN, Mask-RCNN [7] yields promising pixel-level results in instance segmentation. Semantic segmentation requires models to output pixel-wise semantic predictions for a given image. PSPNet [29] introduces spatial pooling at multiple grid scales. To handle scene parsing tasks, UperNet [27] design a more generalized framework based on PSPNet. Image inpainting is a classical problem to fill in the missing regions of the input picture. U-net [24] architecture is popular among recent works [9, 26, 22]. Liu *et al*. [17] introduces partial convolution layer to U-net for alleviating the missing region's influence on the convolution layers, which gets better performance. The classical U-net architecture needs multiple two-times upsample operators in the second half network. Our CARAFE demonstrates universal effectiveness across a wide range of dense prediction tasks.

3. Content-Aware ReAssembly of FEatures

Feature upsampling is a key operator in many modern convolutional network architectures developed for object detection, instance segmentation, and scene parsing, *etc*. In this work, we propose the content-aware reassembly of features (CARAFE) to upsample a feature map. On each location, CARAFE can leverage the underlying content information to predict assembly kernels and assemble the features inside a predefined nearby region. Thanks to the content information, CARAFE can use an adaptive and optimized reassembly kernel in different locations and achieve

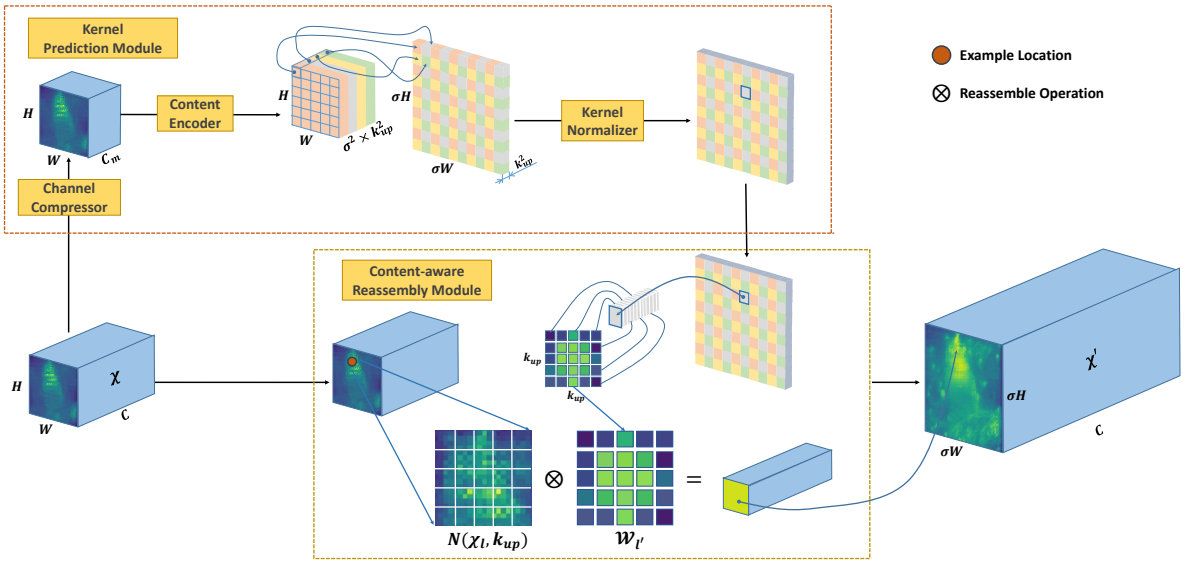

Figure 2: **The overall framework of CARAFE**. CARAFE is composed of two key components, *i.e.*, kernel prediction module and content-aware reassembly module. A feature map with size $C \times H \times W$ is upsampled by a factor of $\sigma(=2)$ in this figure.

better performance than mainstream upsampling operators, *e.g.* bilinear interpolation or deconvolution, *etc*.

3.1. Formulation

CARAFE works as a reassembly operator with content-aware kernels. It consists of two steps. The first step is to predict a reassembly kernel for each target location according to its content, and the second step is to reassemble the features with predicted kernels. Given a feature map $\mathcal{X}$ of size $C \times H \times W$ and an upsample ratio σ (supposing σ is an integer), CARAFE will produce a new feature map $\mathcal{X}'$ of size $C \times \sigma H \times \sigma W$. For any target location $l' = (i', j')$ of the output $\mathcal{X}'$, there is a corresponding source location $l = (i, j)$ at the input $\mathcal{X}$, where $i = \lfloor i'/\sigma \rfloor, j = \lfloor j'/\sigma \rfloor$. Here we denote $N(\mathcal{X}_l, k)$ as the $k \times k$ sub-region of $\mathcal{X}$ centered at the location l, *i.e.*, the neighbor of $\mathcal{X}_l$.

In the first step, the *kernel prediction module* ψ predicts a location-wise kernel $\mathcal{W}_{l'}$ for each location l', based on the neighbor of $\mathcal{X}_l$, as shown in Eqn. 1. The reassembly step is formulated as Eqn. 2, where ϕ is the *content-aware reassembly module* that reassembles the neighbor of $\mathcal{X}_l$ with the kernel $\mathcal{W}_{l'}$:

$$\mathcal{W}_{l'} = \psi(N(\mathcal{X}_l, k_{encoder})). \tag{1}$$

$$\mathcal{X}'_{l'} = \phi(N(\mathcal{X}_l, k_{up}), \mathcal{W}_{l'}). \tag{2}$$

We specify the details of ψ and ϕ in the following parts.

3.2. Kernel Prediction Module

The kernel prediction module is responsible for generating the reassembly kernels in a content-aware manner. Each source location on $\mathcal{X}$ corresponds to σ^2 target locations on $\mathcal{X}'$. Each target locations requires a $k_{up} \times k_{up}$ reassembly kernel, where k_{up} is the reassembly kernel size. Therefore, this module will output the reassembly kernels of size $C_{up} \times H \times W$, where $C_{up} = \sigma^2 k_{up}^2$.

The kernel prediction module is composed of three submodules, *i.e.*, *channel compressor*, *content encoder* and *kernel normalizer*, as shown in Figure 2. The channel compressor reduces the channel of the input feature map. The content encoder then takes the compressed feature map as input and encodes the content to generate reassembly kernels. Lastly, the kernel normalizer applies a softmax function to each reassembly kernel. The three submodules are explained detaily as follows.

Channel Compressor. We adopt a 1×1 convolution layer to compress the input feature channel from C to C_m. Reducing the channel of input feature map leads to less parameters and computational cost in the following steps, making CARAFE more efficient. It is also possible to use larger kernel sizes for the content encoder under the same budget. Experimental results show that reducing the feature channel in an acceptable range will not harm the performance.

Content Encoder. We use a convolution layer of kernel size $k_{encoder}$ to generate reassembly kernels based on the content of input features. The parameters of the encoder is $k_{encoder} \times k_{encoder} \times C_m \times C_{up}$. Intuitively, increasing $k_{encoder}$ can enlarge the receiptive field of the encoder, and exploits the contextual information within a larger region, which is important for predicting the reassembly kernels. However, the computational complexity grows with the square of the kernel size, while the benefits from a larger

kernel size do not. An empirical formula $k_{encoder} = k_{up} - 2$ is a good trade-off between performance and efficiency through our study in Section 5.3.

Kernel Normalizer. Before being applied to the input feature map, each $k_{up} \times k_{up}$ reassembly kernel is normalized with a softmax function spatially. The normalization step forces the sum of kernel values to 1, which is a soft selection across a local region. Due to the kernel normalizer, CARAFE does not perform any rescaling and change the mean values of the feature map, that is why our proposed operator is named the reassembly of features.

3.3. Content-aware Reassembly Module

With each reassembly kernel $\mathcal{W}_{l'}$, the content-aware reassembly module will reassemble the features within a local region via the function ϕ. We adopt a simple form of ϕ which is just a weighted sum operator. For a target location l' and the corresponding square region $N(\mathcal{X}_l, k_{up})$ centered at $l = (i, j)$, the reassembly is shown in Eqn. 3, where $r = \lfloor k_{up}/2 \rfloor$:

$$\mathcal{X}'_{l'} = \sum_{n=-r}^{r} \sum_{m=-r}^{r} \mathcal{W}_{l'(n,m)} \cdot \mathcal{X}_{(i+n,j+m)}. \qquad (3)$$

With the reassembly kernel, each pixel in the region of $N(\mathcal{X}_l, k_{up})$ contributes to the upsampled pixel l' differently, based on the content of features instead of distance of locations. The semantics of the reassembled feature map can be stronger than the original one, since the information from relavant points in a local region can be more attent.

3.4. Relation to Previous Operators

Here we discuss the relations between CARAFE and dynamic filter [12], spatial attention [1], spatial transformer [11] and deformable convolution [4], which share similar design philosophy but with different focuses.

Dynamic Filter. Dynamic filter generates instance-specific convolutional filters conditioned on the input of the network, and then applies the predicted filter on the input. Both dynamic filter and CARAFE are content-aware operators, but a fundamental difference between them lies at their kernel generation process. Specifically, dynamic filter works as a two-step convolution, where the additional filter prediction layer and filtering layer require heavy computation. On the contrary, CARAFE is simply a reassembly of features in local regions, without learning the feature transformation across channels. Supposing the channels of input feature map is C and kernel size of the filter is K, then the predicted kernel parameters for each location is $C \times C \times K \times K$ in dynamic filter. For CARAFE, the kernel parameter is only $K \times K$. Thus, it is more efficient in memory and speed.

Spatial Attention. Spatial attention predicts an attention map with the same size as the input feature, and then rescales the feature map on each location. Our CARAFE reassembles the features in a local region by weighted sum. In summary, spatial attention is a rescaling operator with point-wise guidance while CARAFE is a reassembly operator with region-wise local guidance. Spatial attention can be seen as a special case of CARAFE where the reassembly kernel size is 1, regardless of the kernel normalizer.

Spatial Transformer Networks (STN). STN predicts a global parametric transformation conditioned on the input feature map and warps the feature via the transformation. However, this global parametric transformation assumption is too strong to represent complex spatial variance; and STN is known to be hard to train. Here, CARAFE uses the location-specific reassembly to handle the spatial relations, which enables more flexible local geometry modeling.

Deformable Convolutional Networks (DCN). DCN also adopts the idea of learning geometric transformation and combines it with the regular convolution layers. It predicts kernel offsets other than using grid convolution kernels. Similar to dynamic filter, it is also a heavy parametric operator with 24 times more computational cost than CARAFE. It is also known to be sensitive to parameter initialization.

4. Applications of CARAFE

CARAFE can be seamlessly integrated into existing frameworks where upsampling operators are needed. Here we present some applications in mainstream dense prediction tasks. With negligible additional parameters, CARAFE benefits state-of-the-art methods in both high-level and low-level tasks, such as object detection, instance segmentation, semantic segmentation and image inpainting.

4.1. Object Detection and Instance Segmentation

Feature Pyramid Network (FPN) is an important and effective architecture in the field of object detection and instance segmentation. It significantly improves the performance of popular frameworks like Faster R-CNN and Mask R-CNN. FPN constructs feature pyramids of strong semantics with the top-down pathway and lateral connections. In the top-down pathway, a low-resolution feature map is firstly upsampled by 2x with the nearest neighbor interpolation and then fused with a high-resolution one, as shown in Figure 3.

We propose to substitute the nearest neighbor interpolation in all the feature levels with CARAFE. This modification is smooth and no extra change is required. In addition to the FPN structure, Mask R-CNN adopts a deconvolution layer at the end of mask head. It is used to upsample the predicted digits from 14×14 to 28×28, to obtain finer mask predictions. We can also use CARAFE to replace the deconvolution layer, resulting in even less computational cost.

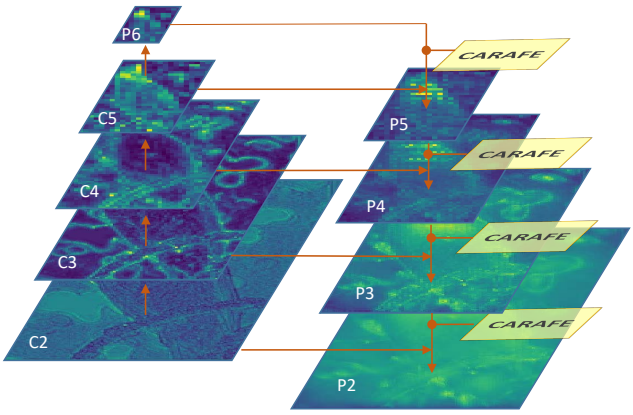

Figure 3: **FPN architecture with CARAFE**. CARAFE upsamples a feature map by a factor of 2 in the top-down pathway. It is integrated into FPN by seamlessly substituting the nearest neighbor interpolation.

4.2. Semantic Segmentation

Semantic segmentation requires the model to output per-pixel level predictions on the whole image, so that high-resolution feature maps are usually preferred. Upsampling is widely adopted to enlarge feature maps and fuse the semantic information of different levels in this task. UperNet is a strong baseline for semantic segmentation. It uses upsampling in the following three components, *i.e.*, PPM, FPN, FUSE. We adopt CARAFE instead of their original upsamplers.

Pyramid Pooling Module (PPM). PPM is the key component in PSPNet which hierarchically down-samples an input feature map into multiple scales $\{1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6\}$, and then upsamples them back to the original sizes with bilinear interpolation. The features are finally fused with the original feature by concatenation. Since the upsampling ratio is very large, we adopt a two-step strategy with CARAFE as a trade-off between performance and efficiency. Firstly we upsamples the $\{1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6\}$ features to half the size of the original feature map with bilinear interpolation, and then use CARAFE to further upsample them by 2x.

Feature Pyramid Network (FPN). Similar to detection models, UperNet also adopts FPN to enrich the feature semantics. It only has 4 different feature levels {P2, P3, P4, P5} with strides {4, 8, 16, 32}. We replace the upsampling operators in the same way as Section 4.1.

Multi-level Feature Fusion (FUSE). UperNet proposes a multi-level feature fusion module after the FPN. It upsamples P3, P4, P5 to the same size as P2 by bilinear interpolation and then fuses these features from different levels by concatenation. The process is equivalent to a sequential upsampling-concatenation which first upsamples P5 to P4 and concatenates them, and then upsamples the concatenated feature map to P3 and so on. We replace the sequential bilinear upsampling here with CARAFE.

4.3. Image Inpainting

The U-net architecture is popular among recent proposed image inpainting methods, such as Global&Local[9] and Partial Conv[17]. There are two upsampling operators in the second half of the network. We simply replace the two upsampling layers with CARAFE and evaluate the performance. As for Partial Conv, we can conveniently keep the mask propagation in CARAFE by updating the mask with our content-aware reassembly kernels.

5. Experiments

5.1. Experimental Settings

Datasets & Evaluation Metrics. We evaluate CARAFE on several important dense prediction benchmarks. We use the *train* split for training and evaluate the performance on the *val* split for all these datasets by default.

Object Detection and Instance Segmentation. We perform experiments on the challenging MS COCO 2017 dataset. Results are evaluated with the standard COCO metric, *i.e.* mAP of IoUs from 0.5 to 0.95.

Semantic Segmentation. We adopt the ADE20k benchmark to evaluate our method in the semantic segmentation task. Results are measured with mean IoU (mIoU) and Pixel Accuracy (P.A.), which respectively indicates the average IoU between predictions and ground truth masks and per-pixel classification accuracy.

Image Inpainting. Places dataset is adopted for image inpainting. We use L1 error (lower is better) and PSNR (higher is better) as evaluation metrics.

Implementation Details. If not otherwise specified, CARAFE adopts a fixed set of hyper-parameters in experiments, where C_m is 64 for the channel compressor and $k_{encoder} = 3$, $k_{up} = 5$ for the content encoder.

Object Detection and Instance Segmentation. We evaluate CARAFE on Faster RCNN and Mask RCNN with the ResNet-50 backbone. FPN is used for these methods. We follow the 1x training schedule and the same hyper-parameters as Detectron [6].

Semantic Segmentation. We use the official implementation of UperNet[2] and adopt the same experiment settings.

Image Inpainting We employ the generator and discriminator networks from Global&Local [9] with a minor modification to achieve better generation quality. Compared to [9], we use only one pathGAN-style [13, 10] discriminator on the inpainted region. We adopt the popular free-form masks [28] for fair comparison. For Paritial Conv [17], we just substitute the convolution layers with the official Partial conv module in our generator.

[2] https://github.com/CSAILVision/semantic-segmentation-pytorch

Table 1: Detection and Instance Segmentation results on MS COCO 2018 *test-dev*.

Method	Backbone	Task	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-50	BBox	36.9	59.1	39.7	21.5	40.0	45.6
Faster R-CNN w/ CARAFE	ResNet-50	BBox	**38.1**	**60.7**	**41.0**	**22.8**	**41.2**	**46.9**
Mask R-CNN	ResNet-50	BBox	37.8	59.7	40.8	22.2	40.7	46.8
	ResNet-50	Segm	34.6	56.5	36.8	18.7	37.3	45.1
Mask R-CNN w/ CARAFE	ResNet-50	BBox	**38.8**	**61.2**	**42.1**	**23.2**	**41.7**	**47.9**
	ResNet-50	Segm	**35.9**	**58.1**	**38.2**	**19.8**	**38.6**	**46.5**

Table 2: Detection results with Faster RCNN. Various upsampling methods are used in FPN.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FLOPs
Nearest	36.5	58.4	39.3	21.3	40.3	47.2	0
Bilinear	36.7	58.7	39.7	21.0	40.5	47.5	8k
Nearest + Conv	36.6	58.6	39.5	21.4	40.3	46.4	4.7M
Bilinear + Conv	36.6	58.7	39.4	21.6	40.6	46.8	4.7M
Deconv [21]	36.4	58.2	39.2	21.3	39.9	46.5	1.2M
Pixel Shuffle[25]	36.5	58.8	39.1	20.9	40.4	46.7	4.7M
GUM[18]	36.9	58.9	39.7	21.5	40.6	48.1	1.1M
S.A.[1]	36.9	58.8	39.8	21.7	40.8	47.0	28k
CARAFE	**37.8**	**60.1**	**40.8**	**23.1**	**41.7**	**48.5**	199k

Table 3: Instance Segmentation results with Mask RCNN. Various upsampling methods are used in mask head.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Nearest	32.7	55.0	34.8	17.7	35.9	44.4
Bilinear	34.2	55.9	36.4	18.5	37.5	46.2
Deconv	34.2	55.5	36.3	17.6	37.8	46.7
Pixel Shuffle	34.4	56.0	36.6	**18.5**	37.6	47.5
GUM	34.3	55.7	36.5	17.6	37.6	46.9
S.A.	34.1	55.6	36.5	17.6	37.4	46.6
CARAFE	**34.7**	**56.2**	**37.1**	18.2	**37.9**	**47.5**

5.2. Benchmarking Results

Object Detection & Instance Segmentation. We first evaluate our method by substituting the nearest neighbor interpolation in FPN with CARAFE for both Faster RCNN and Mask RCNN, and the deconvolution layer in the mask head for Mask RCNN. As shown in Table 1, CARAFE improves Faster RCNN by 1.2% on bbox AP, and Mask RCNN by 1.3% on mask AP. The improvements of AP_S, AP_M, AP_L are all above 1% AP, which suggests that it is benefitial for various object scales.

Our encouraging performance is supported by the qualitative results as shown in Figure 1. We visualize the feature maps in the top-down pathway of FPN and compare CARAFE with the baseline, *i.e.*, nearest neighbor interpolation. It is obvious that with the content-aware reassembly, the feature maps are more discriminative and a more accurate mask for the object is predicted. In Figure 4, we show some examples of instance segmentation results comparing the baseline and CARAFE.

To investigate the effectiveness of different upsampling operators, we perform extensive experiments on Faster RCNN by using different operators to perform upsampling in FPN. Results are illustrated in Table 2. For 'Nearest + Conv' and 'Bilinear + Conv', we add an extra 3×3 convolution layer after upsampling the feature map by the corresponding interpolation. 'Deconv', 'Pixel Shuffle', 'GUM' are three representative learning based upsampling methods. We also compare spatial attention here, indicated as 'S.A.'. CARAFE achieves the best AP among all these upsampling operators, and the FLOPs is relatively small, which proves it is both effective and efficient. The results of 'Nearest + Conv' and 'Bilinear + Conv' show that extra parameters do not lead to a significant gain. 'Deconv', 'Pixel Shuffle', 'GUM' and 'S.A.' obtain inferior performance to CARAFE, indicating that the design of effective upsampling operators is nontrivial.

Besides FPN which is a pyramid feature fusion structure, we also explore different upsampling operators in the mask head. In typical Mask R-CNN, a deconvolution layer is adopted to upsample the RoI features by 2x. For fair comparison, we do not make any changes to FPN, and only replace the deconvolution layer with various operators. Since we only modify the mask prediction branch, performance is reported in terms of mask AP, as shown in Table 3. CARAFE achieves the best performance in instance segmentation among these methods.

In Table 4, we report the object detection and instance segmentation results of adopting CARAFE in FPN and mask head on Mask RCNN respectively. Consistent improvements are achieved in these experiments.

Semantic Segmentation. We replace the upsamplers in UperNet with CARAFE and evaluate the results on ADE20k benchmark. As shown in Table 5, CARAFE improves the mIoU by a large margin from 40.44% to 42.23% with single scale testing. Note that UperNet with CARAFE also achieves better performance than recent

Figure 4: Comparison of instance segmentation results between baseline (top row) and CARAFE (bottom row) on COCO 2017 val.

Table 4: Detection and Instance Segmentation results with Mask RCNN via adopting CARAFE in FPN and mask head respectively. M.H. indicates using CARAFE in mask head.

FPN	M.H.	Task	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
		Bbox	37.4	59.1	40.3	21.2	41.2	48.5
		Segm	34.2	55.5	36.3	17.6	37.8	46.7
✓		Bbox	38.6	60.7	**42.2**	23.2	42.1	49.5
		Segm	35.2	57.2	37.5	19.3	38.3	47.6
	✓	Bbox	37.3	59.0	40.2	21.8	40.8	48.6
		Segm	34.7	56.2	37.1	18.2	37.9	47.5
✓	✓	Bbox	**38.6**	**60.9**	41.9	**23.4**	**42.3**	**49.8**
		Segm	**35.7**	**57.6**	**38.1**	**19.4**	**39.0**	**48.7**

Table 5: Semantic Segmentation results on ADE20k val. Single scale testing is used in our experiments.

Method	Backbone	mIoU	P.A.
PSPNet	ResNet-50	41.68	80.04
PSANet	ResNet-50	41.92	80.17
UperNet[3]	ResNet-50	40.44	79.80
UperNet w/ CARAFE	ResNet-50	**42.23**	**80.34**

strong baselines such as PSPNet[29] and PSANet[30].

We perform a step-by-step study to inspect the effectiveness of modifying different components in UperNet, as described in Section 4.2. Results in Table 6 show that CARAFE is helpful for all the three components and the combination of them results in further gains.

Image Inpainting. We show that CARAFE is also effective in low-level tasks such as image inpainting. By replacing the upsampling operators with CARAFE in two strong baselines Global&Local [9] and Partial Conv [17], we observe significant improvements for both methods. As shown in Table 7, our method improves two baselines by 1.1 db and 0.2 db on the PSNR metric.

[3]We report the performance in model zoo of the official implementation.

Table 6: Effects of adopting CARAFE in each component of UperNet.

PPM	FPN	FUSE	mIoU	P.A.
✓			40.85	79.97
	✓		40.79	80.01
		✓	41.06	80.23
✓	✓		41.55	80.30
✓		✓	42.01	80.11
	✓	✓	41.93	**80.34**
✓	✓	✓	**42.23**	**80.34**

Table 7: Image inpainting results on Places val.

Method	L1(%)	PSNR(dB)
Global&Local	6.78	19.58
Partial Conv	5.96	20.78
Global&Local w/ CARAFE	6.00	20.71
Partial Conv w/ CARAFE	**5.72**	**20.98**

5.3. Ablation Study & Further Analysis

Model Design & Hyper-parameters. We investigate the influence of hyper-parameters in the model design, which are the compressed channels C_m, encoder kernel size $k_{encoder}$ and reassembly kernel size k_{up}. We also test different normalization methods in the kernel normalizer. We perform the ablation study of the designs and settings on Faster RCNN with a ResNet-50 backbone, and evaluate the results on COCO 2017 val.

Towards an efficient design, we first analyze the computational complexity measured by FLOPs. When upsampling the feature map with input channel C_{in} by a factor of σ, the per pixel FLOPs of CARAFE is computed as $2(C_{in}+1)C_m + 2(C_m k_{encoder}^2 + 1)\sigma^2 k_{up}^2 + 2\sigma^2 k_{up}^2 C_{in}$, referring to [19].

We experiment with different values of C_m in the channel compressor. In addition, we also try removing the channel compressor module, which means the content encoder directly uses input features to predict reassembly kernels.

Figure 5: CARAFE performs content-aware reassembly when upsampling a feature map. Red units are reassembled into the green center unit by CARAFE in the top-down pathway of a FPN structure.

Table 8: Ablation study of various compressed channels C_m. N/A means *channel compressor* is removed.

C_m	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
16	37.6	60.1	40.6	22.7	41.6	48.4
32	37.7	60.3	40.7	22.8	41.2	49.0
64	**37.8**	60.1	40.8	23.1	41.7	48.5
128	**37.8**	60.1	40.8	22.4	41.7	48.7
256	**37.8**	60.4	40.8	22.7	41.3	48.8
N/A	**37.8**	60.3	40.8	22.9	41.5	48.7

Experimental results in Table 8 show that compress C_m down to 64 leads to no performance decline, while being more effecient. A further smaller C_m will result in a slightly drop of the performance. With no channel compressor, it can achieve the same performance, which proves that the channel compressor is just used for speeding up. Based on the above results, we set C_m to 64 by default as a trade-off between performance and efficiency.

We then investigate the influence of $k_{encoder}$ and k_{up}. Intuitively, increasing k_{up} also requires a larger $k_{encoder}$, since the content encoder needs a large receptive field to predict a large reassembly kernel. As illustrated in Table 9, increasing $k_{encoder}$ and k_{up} at the same time can boost the performance, while just enlarging one of them will not. We summarize an empirical formula that $k_{encoder} = k_{up} - 2$, which is a good choice in all the settings. Though adopting larger kernel size is shown helpful, we set $k_{up} = 5$ and $k_{encoder} = 3$ by default as a tradeoff between performance and efficiency.

Other than the softmax function, we also test other alternatives in the kernel normalizer, such as sigmoid or sigmoid with normalization. As shown in Table 10, 'Softmax' and 'Sigmoid Normalized' have the same performance and better than 'Sigmoid', which shows that it is crucial to normalize the reassembly kernel to be summed to 1.

How CARAFE Works. We conduct further qualitative study to figure out how CARAFE works. With a trained Mask RCNN model adopting CARAFE as the upsampling operator, we visualize the reassembling process in Figure 5. In the FPN structure, the low-resolution feature map will be consecutively upsampled for several times to a higher resolution, so a pixel in the upsampled feature map reassembles information from a more larger region. We sample some pixels in the high-resolution feature map, and see which neighbors it is reassembled from. The green circle denotes example locations and red dots indicates highly weighted sources during the reassembly. From the figure, we can clearly learn that CARAFE is content-aware. It tends to reassemble points with similar semantic information. A location at human body prefers other points from the same human, rather than other objects or nearby background. For locations in the background regions which has weaker semantics, the reassembly is more uniform or just biased on points with similar low-level texture features.

Table 9: Detection results with various encoder kernel size $k_{encoder}$ and reassembly kernel size k_{up}.

$k_{encoder}$	k_{up}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	3	37.3	59.6	40.5	22.0	40.7	48.1
1	5	37.3	59.9	40.0	22.3	41.1	47.3
3	3	37.3	59.7	40.4	22.1	40.8	48.3
3	5	37.8	60.1	40.8	23.1	41.7	48.5
3	7	37.7	60.0	40.9	23.0	41.5	48.4
5	5	37.8	60.2	40.7	22.5	41.4	48.6
5	7	**38.1**	60.4	41.3	23.0	41.6	48.8
7	7	38.0	60.2	41.1	23.0	41.8	48.8

Table 10: Ablation study of different normalization methods in *kernel normalizer*.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Sigmoid	37.4	59.8	40.2	23.1	40.9	47.4
Sigmoid Normalize	**37.8**	60.1	40.7	22.6	41.6	48.0
Softmax	**37.8**	60.1	40.8	23.1	41.7	48.5

6. Conclusion

In this work, we propose Content-Aware ReAssembly of FEatures (CARAFE), a universal, lightweight and highly effective upsampling operator. It consistently boosts the performances on standard benchmarks in object detection, instance/semantic segmentation and inpainting by 1.2%, 1.3%, 1.8% and 1.1db respectively. More importantly, CARAFE introduces little computational overhead and can be readily integrated into modern network architectures. Future directions include exploring the applicability of CARAFE in low-level vision tasks such as image restoration and super-resolution.

References

[1] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 4, 6

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018. 11

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision*, 2018. 1

[4] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE International Conference on Computer Vision*, 2017. 4

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016. 1

[6] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 5, 11

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017. 2, 11, 13

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 11

[9] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):107, 2017. 1, 2, 5, 7, 11, 15

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5

[11] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. 4

[12] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, 2016. 4

[13] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, 2016. 5, 11

[14] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2017. 1

[15] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017. 1, 2, 11

[16] 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 *European Conference on Computer Vision*, 2014. 2, 11

[17] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision*, 2018. 2, 5, 7

[18] Davide Mazzini. Guided upsampling network for real-time semantic segmentation, 2018. 2, 6

[19] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 7

[20] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, 2016. 1

[21] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *IEEE International Conference on Computer Vision*, Dec 2015. 1, 2, 6

[22] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 2, 11

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 1, 2

[25] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2016. 2, 6

[26] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[27] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision*, 2018. 2, 11, 14

[28] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589*, 2018. 5, 11

[29] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 7

[30] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *European Conference on Computer Vision*, 2018. 7

[31] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2, 11

[32] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 11

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 2018. 2

Appendix A. Detail Experimental Settings

Object Detection and Instance Segmentation. We evaluate CARAFE on Faster RCNN [23] and Mask RCNN [7] with the ResNet-50 backbone [8]. FPN [15] is used for these methods. In both training and inference, we resize an input image such that its shorter edge has 800 pixels or longer edge has 1333 pixels without changing its aspect ratio. We adopt synchronized SGD with an initial learning rate of 0.02, a momentum of 0.9 and a weight decay of 0.0001. We use batchsize of 16 over 8 GPUs (2 images per GPU). Following the 1x training schedule as Detectron [6], we train 12 epochs in total and decrease the learning rate by a factor of 0.1 at epoch 8 and 11.

Semantic Segmentation. We use the official implementation of UperNet [27] with the ResNet-50 backbone. During the training, an input image is resized such that the size of its shorter edge is randomly selected from $\{300, 375, 450, 525, 600\}$. In inference, we apply the single scale testing for fair comparison and the shorter edge of an image is set to 450 pixels. The maximum length of the longer edge of an image is set to 1200 in both training and inference. We adopt synchronized SGD with an initial learning rate of 0.02, a momentum of 0.9 and a weight decay of 0.0001. We use batchsize of 16 over 8 GPUs (2 images per GPU), and synchronized batch normalization is adopted as a common practice in semantic segmentation. Following [2], the 'poly' learning rate policy in which the learning rate of current iteration equals to the initial learning rate multiplying $(1 - iter/max\_iter)^{power}$ is adopted. We set power to 0.9 and train 20 epochs in total.

Image Inpainting. We employ the generator and discriminator networks from Global&Local [9] as the baseline. Our generator takes a 256×256 image $\mathbf{x}$ with masked region M as input and produces a 256×256 prediction of the missing region $\hat{\mathbf{y}}$ as output. Then we combine the predicted image with the input by $\mathbf{y} = (1 - M) \odot \mathbf{x} + M \odot \hat{\mathbf{y}}$. Finaly, the combined output $\mathbf{y}$ is fed into the discriminator. We apply a simple modification to the baseline model to achieve better generation quality. Compared to the original model that employs two discriminators, we employ only one patchGAN-style discriminator[13] on the inpainted region. This modification can achieve better image quality.

For fair comparison and taking real-world application into consideration, we use the free-form mask introduced by [28] as the binary mask M. During training, Adam solver with learning rate $1e^{-4}$ is adopted where $\beta_1 = 0.5$ and $\beta_2 = 0.9$. Training batch size is 32. The input and output are linearly scaled within range $[-1, 1]$.

Appendix B. More Visualizations of CARAFE

We demonstrate how CARAFE performs content-aware reassembly with more examples in Figure 6. Red units are reassembled into the green center unit by CARAFE in the top-down pathway of a FPN structure.

Appendix C. More Visual Results Comparison

Object Detection and Instance Segmentation. As illustrated in Figure 7, we provide more object detection and instance segmentation results comparison between Mask RCNN basline and Mask RCNN with CARAFE on COCO [16] 2017 val.

Semantic Segmentation. We compare the semantic segmentation results between UperNet baseline and UperNet with CARAFE on ADE20k [32] val in Figure 8.

Image Inpainting. Comparison of image inpainting results between Gobal&Local baseline and Gobal&Local with CARAFE on Places[31] val is shown in Figure 9.

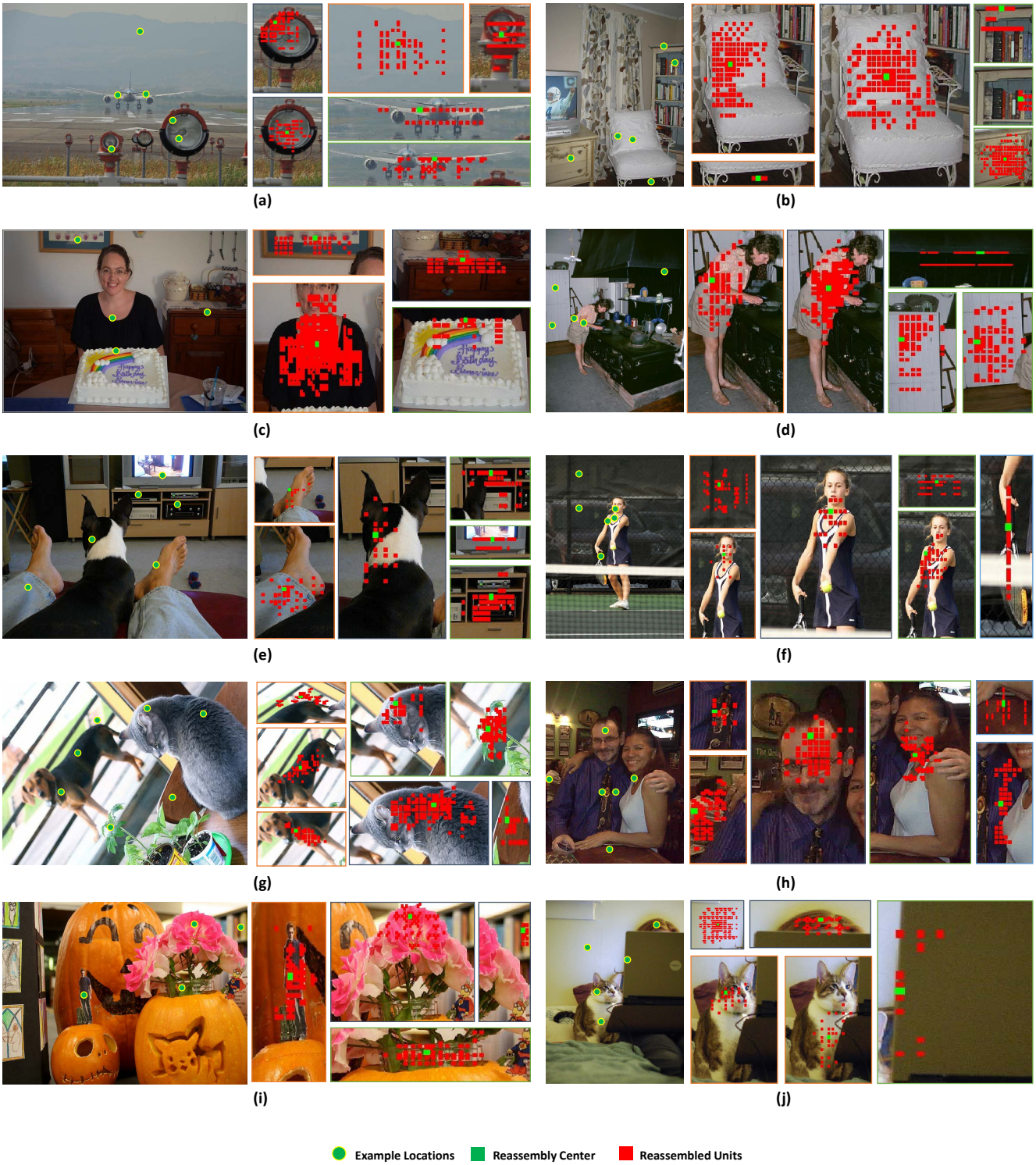

Figure 6: CARAFE performs content-aware reassembly when upsampling a feature map. Red units are reassembled into the green center unit by CARAFE in the top-down pathway of a FPN structure.

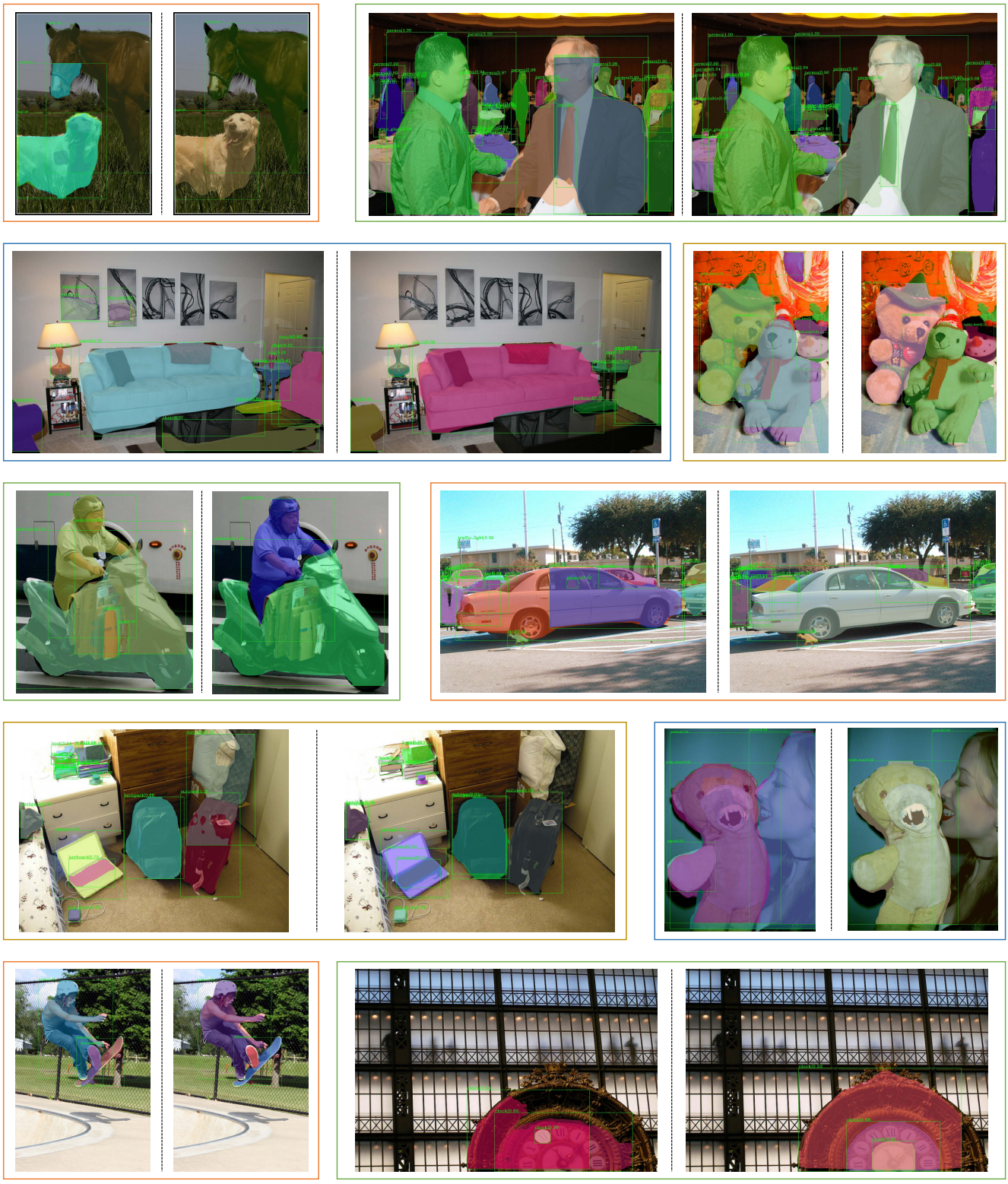

Figure 7: More comparison of object detection and instance segmentation results between Mask RCNN [7] baseline (left to the dash line) and Mask RCNN with CARAFE (right to the dash line) on COCO 2017 val.

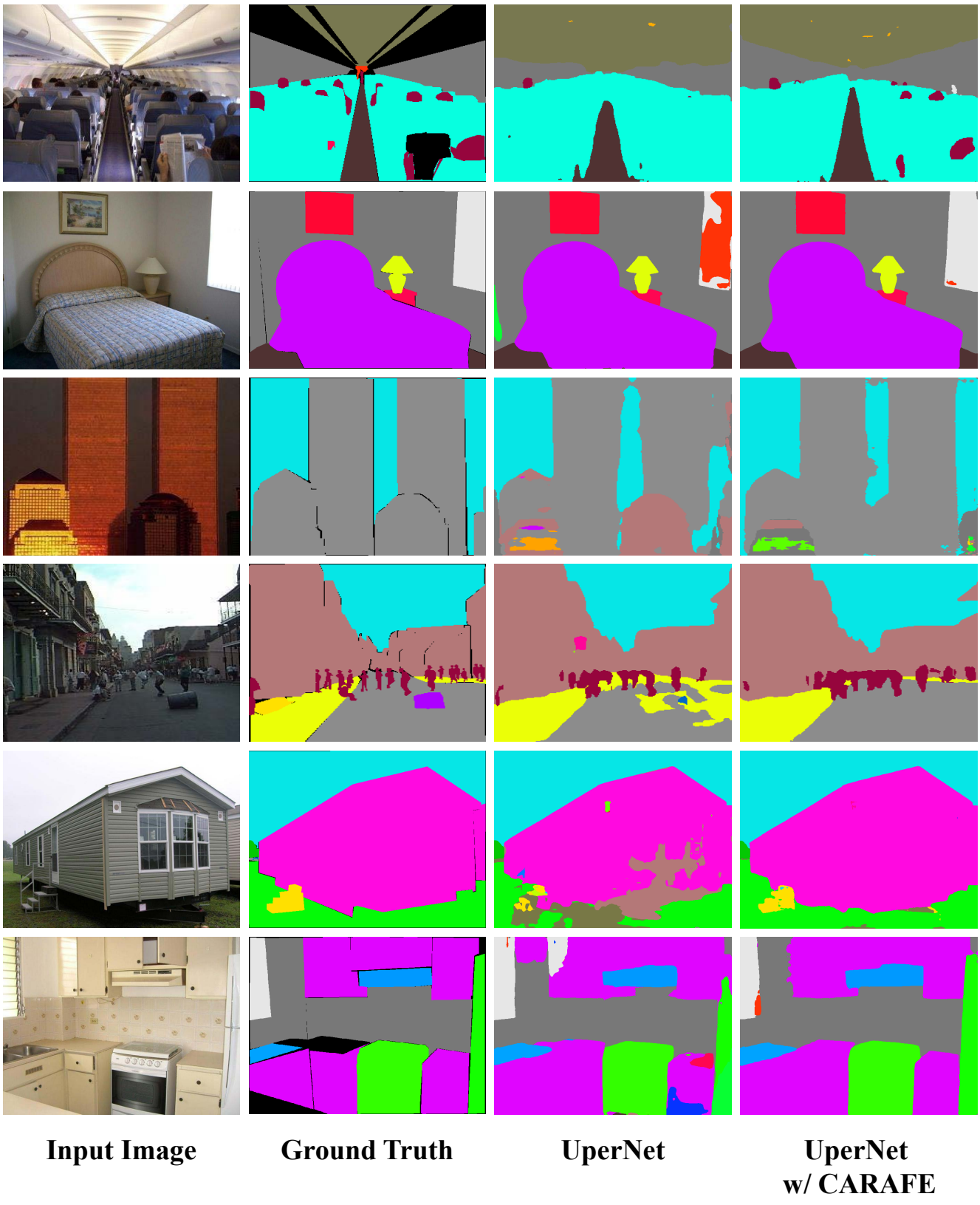

Figure 8: Comparison of semantic segmentation results between UperNet [27] baseline and UperNet with CARAFE on ADE20k val. Columns from left to right correspond to the input image, ground truth, baseline results and CARAFE results respectively.

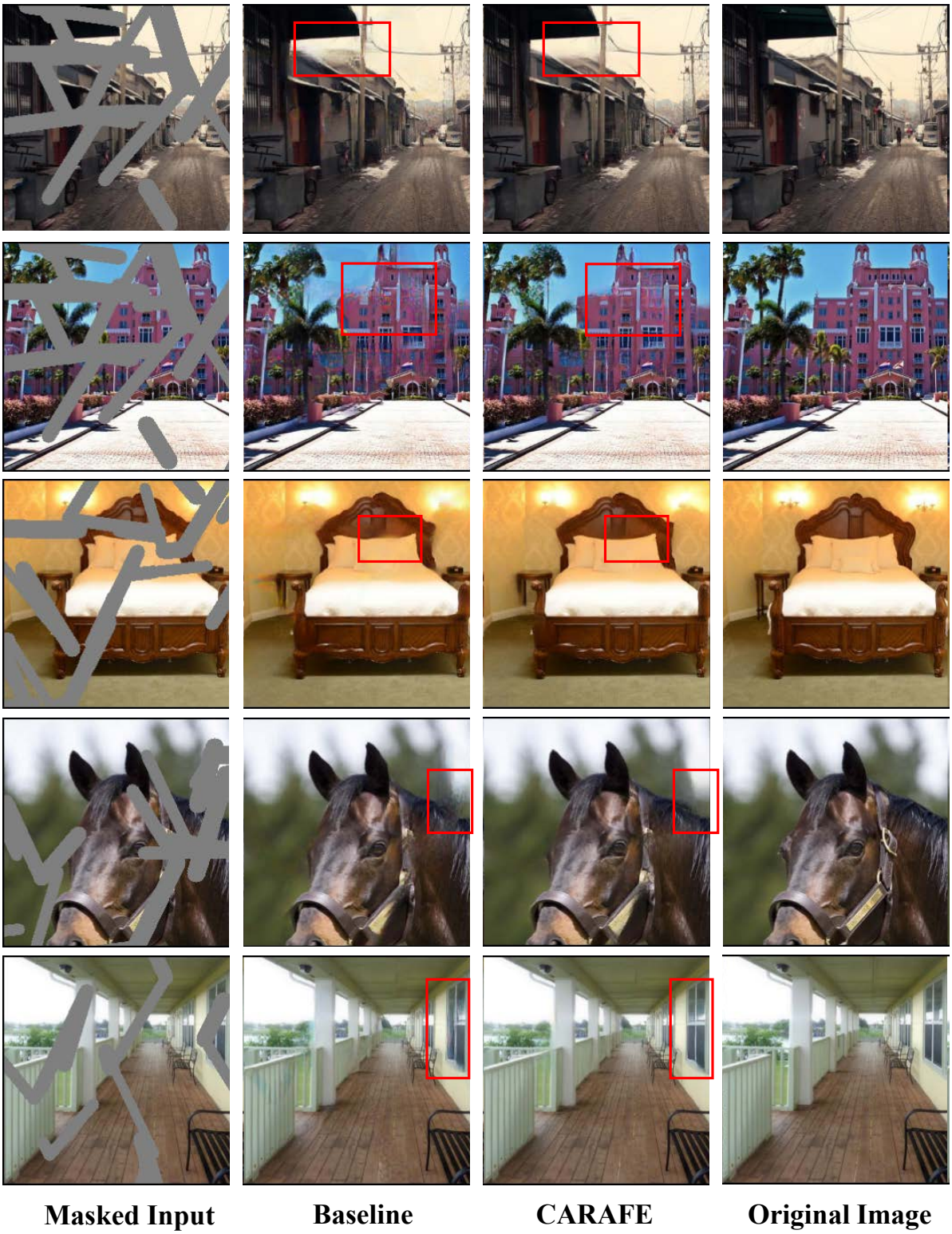

Figure 9: Comparison of image inpainting results between Global&Local [9] baseline and Global&Local with CARAFE on Places val. Columns from left to right correspond to the masked input, baseline results, CARAFE results and original image respectively.