CARAFE: Content-Aware ReAssembly of FEatures

Jiaqi Wang
Kai Chen
Rui Xu
Ziwei Liu
Chen Change Loy
Dahua Lin

The Chinese University of Hong Kong
Nanyang Technological University

ICCV 2019, Oral
Results on COCO Challenge 2019

Best score without external data

Comparison of our approach (MMDet) with 2018 winning entries on COCO test-dev.

MASK AP

<table>
<thead>
<tr>
<th></th>
<th>2018 winner (single model)</th>
<th>2018 winner</th>
<th>Single model</th>
<th>Final results</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.474</td>
<td>0.49</td>
<td>0.494</td>
<td>0.513</td>
<td></td>
</tr>
</tbody>
</table>

BOX AP

<table>
<thead>
<tr>
<th></th>
<th>2018 winner (single model)</th>
<th>2018 winner</th>
<th>Single model</th>
<th>Final results</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.541</td>
<td>0.56</td>
<td>0.56</td>
<td>0.578</td>
<td></td>
</tr>
</tbody>
</table>
Results on COCO
Results on COCO
Results on COCO
Feature Upsampling

Upsample with a $\sigma$ ratio
Feature Upsampling

FPN (Object Detection)

Feature upsampling is used in many different deep learning architectures
The design of upsampling operator is critical for dense prediction tasks.
Existing Upsampling Operators

**NN/Bilinear Interpolations**
Leverage distances to measure the correlations between pixels

**Deconvolution (Transposed Convolution)**
An inverse operator of a convolution

**Pixel Shuffle**
Reshapes depth on the channel space into width and height on the spatial space

- Hand-crafted upsampling kernel
  - Exploit sub-pixel neighbors

- Shared upsampling kernel across different locations
  - Limited kernel size

- Content-unrelated kernel
  - Limited field of view
**Desired Properties**

**Desired properties of an upsampling operator**

- Content-aware
- Large field of view
- Lightweight and fast to compute

Upsample with a $\sigma$ ratio
Content-Aware ReAssembly of FEatures (CARAFE)

A universal, lightweight and highly effective upsampling operator

- **Content-Aware**
  - Generates adaptive and content-aware kernels for upsampling on-the-fly

- **Large Field of View**
  - Aggregate contextual information within a large receptive field

- **Lightweight and Efficient**
  - Introduces little computational overhead

- **General**
  - Can be integrated into various network architectures
Key Idea: Feature Reassembly

Input feature map

Local region of the input feature map

Soft selection performed by a reassembly kernel

Content-aware reassembly by weighted sum

A target pixel of the upsample feature map
Key Idea: Feature Reassembly

- Content-aware reassembly by weighted sum
- Local region of the input feature map
- Soft selection performed by a reassembly kernel
- A target pixel of the upsample feature map

- Content-aware reassembly kernels for each target location
- Large kernel size to ensure large view field
Key Idea: Feature Reassembly

1. Content-aware and efficient

2. Extract local neighborhood

3. Content-aware location-wise reassembly kernels prediction

4. Reassemble local features by kernels
Kernel Prediction

- **Goal:** Generate reassembly kernels in a content-aware manner

- **Input:** feature map with a size of $C \times H \times W$

- **Output:** a collection of kernels arranged in a tensor of shape

$$k_{up}^2 \times \sigma H \times \sigma W$$

$k_{up}^2 = \text{size of a local region in the input feature map}$

$\sigma = \text{unsampling ratio}$
Kernel Prediction

A collection of reassembly kernels

A reassembly kernel

Extract corresponding reassembly kernel

\( k_{up} \)
Kernel Prediction

Channel compression

• $1 \times 1$ convolution layer to compress the number of input feature channels from $C$ to $C_m$
• Reduces computational cost for subsequent steps
Kernel Prediction

Content encoding
• Takes the compressed feature map as input
• Encodes the content into a tensor of shape $\sigma^2 k_{up}^2 \times H \times W$
Kernel Prediction

Content encoding

• Takes the compressed feature map as input
• Encodes the content into a tensor of shape \( \sigma^2 k_{up}^2 \times H \times W \)
• Reshapes the tensor to \( k_{up}^2 \times \sigma H \times \sigma W \) (like pixel shuffling)
Kernel Prediction

Kernel normalization

• Each reassembly kernel is normalized with a softmax function
• Kernel values sum to one, thus performing soft selection across a local region
• Neither perform rescaling nor change the mean values of the feature map
Content-Aware Reassembly

Extract $k_{up} \times k_{up}$ local region given a location

Extract the corresponding reassembly kernel
Content-Aware Reassembly

Reassemble the local feature with the reassembly kernel
Perform weighted sum to get the features of target location in the upsampled feature map
Repeat for each target location to obtain an upsampled feature map
Overall Pipeline

Kernel Prediction

Channel Compressor $H \rightarrow W_{in}$

Content Encoder $H \rightarrow W \\ W \times k^2_{up}$

Kernel Normalizer $\sigma W \rightarrow \sigma W \\ k^2_{up}$

Collection of reassembly kernels

Content-Aware Reassembly

Input feature maps $H \rightarrow W \rightarrow C$

Local feature $k_{up}$

Reassembly kernel $k_{up}$

Reassembled features for a target pixel $\sigma H' \rightarrow \sigma W \rightarrow C$
How It Works

In FPN, a low-resolution feature map will be consecutively upsampled for several times to a higher resolution one.

A pixel in the upsampled feature map reassembles information from a larger region.

CARAFE tends to capture information according to the shape of the object.

The final feature map is more discriminative thus more accurate masks can be predicted.
How It Works

Location

Reassembly center

Highly weighted neighbors reassembled by CARAFE. These points have similar semantic information to the reassembly center
How It Works

Location

Reassembly center

Highly weighted neighbors reassembled by CARAFE. These points have similar semantic information to the reassembly center
Substituting the nearest neighbour interpolation in the top-down pathway of Feature Pyramid Network (FPN)

From naïve up-sampling to content-aware up-sampling

Using FPN with CARAFE in Faster R-CNN and Mask R-CNN
Object Detection and Instance Segmentation

Replacing the transposed convolution layer in the mask head with CARAFE

Mask Head in Mask RCNN
Object Detection and Instance Segmentation

Results on COCO2017 test-dev

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>Task</th>
<th>AP</th>
<th>AP_S</th>
<th>AP_M</th>
<th>AP_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>ResNet-50</td>
<td>BBox</td>
<td>36.9</td>
<td>21.5</td>
<td>40.0</td>
<td>45.6</td>
</tr>
<tr>
<td>Faster R-CNN w/ CARAFE</td>
<td>ResNet-50</td>
<td>BBox</td>
<td><strong>38.1</strong></td>
<td><strong>22.8</strong></td>
<td><strong>41.2</strong></td>
<td><strong>46.9</strong></td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-50</td>
<td>BBox</td>
<td>37.8</td>
<td>22.2</td>
<td>40.7</td>
<td>46.8</td>
</tr>
<tr>
<td></td>
<td>ResNet-50</td>
<td>Segm</td>
<td>34.6</td>
<td>18.7</td>
<td>37.3</td>
<td>45.1</td>
</tr>
<tr>
<td>Mask R-CNN w/ CARAFE</td>
<td>ResNet-50</td>
<td>BBox</td>
<td><strong>38.8</strong></td>
<td><strong>23.2</strong></td>
<td><strong>41.7</strong></td>
<td><strong>47.9</strong></td>
</tr>
<tr>
<td></td>
<td>ResNet-50</td>
<td>Segm</td>
<td><strong>35.9</strong></td>
<td><strong>19.8</strong></td>
<td><strong>38.6</strong></td>
<td><strong>46.5</strong></td>
</tr>
</tbody>
</table>
Object Detection and Instance Segmentation

Mask R-CNN

Mask R-CNN with CARAFE
Object Detection results with Faster RCNN.
Comparing different upsampling methods in FPN

<table>
<thead>
<tr>
<th>Methods</th>
<th>AP</th>
<th>APₜ</th>
<th>APₓ</th>
<th>APₙ</th>
<th>FLOPs</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest</td>
<td>36.5</td>
<td>21.3</td>
<td>40.3</td>
<td>47.2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bilinear</td>
<td>36.7</td>
<td>21.0</td>
<td>40.5</td>
<td>47.5</td>
<td>8k</td>
<td>0</td>
</tr>
<tr>
<td>Nearest + Conv</td>
<td>36.6</td>
<td>21.4</td>
<td>40.3</td>
<td>46.4</td>
<td>4.7M</td>
<td>590k</td>
</tr>
<tr>
<td>Bilinear + Conv</td>
<td>36.6</td>
<td>21.6</td>
<td>40.6</td>
<td>46.8</td>
<td>4.7M</td>
<td>590k</td>
</tr>
<tr>
<td>Deconv</td>
<td>36.4</td>
<td>21.3</td>
<td>39.9</td>
<td>46.5</td>
<td>1.2M</td>
<td>590k</td>
</tr>
<tr>
<td>Pixel Shuffle</td>
<td>36.5</td>
<td>20.9</td>
<td>40.4</td>
<td>46.7</td>
<td>4.7M</td>
<td>2.4M</td>
</tr>
<tr>
<td>GUM</td>
<td>36.9</td>
<td>21.5</td>
<td>40.6</td>
<td>48.1</td>
<td>1.1M</td>
<td>132k</td>
</tr>
<tr>
<td>Spatial Attention</td>
<td>36.9</td>
<td>21.7</td>
<td>40.8</td>
<td>47.0</td>
<td>28k</td>
<td>2.3k</td>
</tr>
<tr>
<td>CARAFE</td>
<td><strong>37.8</strong></td>
<td><strong>23.1</strong></td>
<td><strong>41.7</strong></td>
<td><strong>48.5</strong></td>
<td>199k</td>
<td>74k</td>
</tr>
</tbody>
</table>

Semantic Segmentation

Pyramid Pooling Module (PPM)

Feature Pyramid Network (FPN)

Multi-level Feature Fusion (FUSE)

[1] T. Xiao et al., Unified perceptual parsing for scene understanding, ECCV 2018
Semantic Segmentation

Results on ADE20k val
Single-scale testing is used in our experiments

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>mIoU</th>
<th>Pixel Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPNet</td>
<td>ResNet-50</td>
<td>41.68</td>
<td>80.04</td>
</tr>
<tr>
<td>PSANet</td>
<td>ResNet-50</td>
<td>41.92</td>
<td>80.17</td>
</tr>
<tr>
<td>UperNet</td>
<td>ResNet-50</td>
<td>40.44</td>
<td>79.80</td>
</tr>
<tr>
<td><strong>UperNet w/ CARAFE</strong></td>
<td><strong>ResNet-50</strong></td>
<td><strong>42.23</strong></td>
<td><strong>80.34</strong></td>
</tr>
</tbody>
</table>

Effect of CARAFE on each component

<table>
<thead>
<tr>
<th>Component</th>
<th>mIoU</th>
<th>Pixel Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPM</td>
<td>✓</td>
<td>40.85</td>
</tr>
<tr>
<td>FPN</td>
<td>✓</td>
<td>40.97</td>
</tr>
<tr>
<td><strong>FUSE</strong></td>
<td>✓</td>
<td>41.06</td>
</tr>
<tr>
<td>PPM</td>
<td>✓ ✓</td>
<td>41.55</td>
</tr>
<tr>
<td>FPN</td>
<td>✓ ✓</td>
<td>42.01</td>
</tr>
<tr>
<td>FUSE</td>
<td>✓ ✓ ✓</td>
<td><strong>42.23</strong></td>
</tr>
</tbody>
</table>

- Pyramid Pooling Module (PPM)
- Feature Pyramid Network (FPN)
- Multi-level Feature Fusion (FUSE)

[1] T. Xiao et al., Unified perceptual parsing for scene understanding, ECCV 2018
Semantic Segmentation

Input

Ground-truth

UperNet

UperNet with CARAFE

Input

Ground-truth

UperNet

UperNet with CARAFE
Image Inpainting

[1] S. Iizuka et al., Globally and locally consistent image completion, ACM ToG 2017
## Image Inpainting

Results on **Places val.**

<table>
<thead>
<tr>
<th>Methods</th>
<th>L1(%)</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global&amp;Local [1]</td>
<td>6.78</td>
<td>19.05</td>
</tr>
<tr>
<td>Partial Conv [2]</td>
<td>5.96</td>
<td>20.78</td>
</tr>
<tr>
<td>Global&amp;Local w/ CARAFE</td>
<td>6.00</td>
<td>20.71</td>
</tr>
<tr>
<td>Partial Conv w/ CARAFE</td>
<td>5.72</td>
<td>20.98</td>
</tr>
</tbody>
</table>

[1] S. Iizuka et al., Globally and locally consistent image completion, ACM ToG 2017
## Image Inpainting

### Results on Places val.

<table>
<thead>
<tr>
<th>Methods</th>
<th>L1(%)</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global&amp;Local [1]</td>
<td>6.78</td>
<td>19.05</td>
</tr>
<tr>
<td>Partial Conv [2]</td>
<td>5.96</td>
<td>20.78</td>
</tr>
<tr>
<td>Global&amp;Local w/ CARAFE</td>
<td>6.00</td>
<td>20.71</td>
</tr>
<tr>
<td>Partial Conv w/ CARAFE</td>
<td><strong>5.72</strong></td>
<td><strong>20.98</strong></td>
</tr>
</tbody>
</table>

[1] S. Iizuka et al., Globally and locally consistent image completion, ACM ToG 2017

**Content-Aware ReAssembly of FEatures (CARAFE)**

A universal, lightweight and highly effective upsampling operator

<table>
<thead>
<tr>
<th>Content-Aware</th>
<th>Large Field of View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generates adaptive and content-aware kernels for upsampling on-the-fly</td>
<td>Aggregate contextual information within a large receptive field</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lightweight and Efficient</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduces little computational overhead</td>
<td>Can be integrated into various network architectures</td>
</tr>
</tbody>
</table>
One more thing
MMDetection: Supported Methods

- RPN
- Guided Anchoring
- Fast / Faster R-CNN
- R-FCN
- Grid R-CNN
- Libra R-CNN
- Mask R-CNN
- Mask scoring R-CNN
- Double Head R-CNN
- Cascade R-CNN
- Hybrid Task Cascade
- NAS-FPN
- M2Det
- HRNet
- ScratchDet
- Mixed Precision Training
- CARAFE
- DCN / DCN V2
- Weight Standardization
- Generalized Attention
- GCNet
- RetinaNet
- SSD
- GHM
- OHEM
- FCOS
- FoveaBox
- Reppoints

GitHub: MMDet
MMDetection

2.1k forks
6.9k stars
70 contributors
329 merged/closed pull requests
772 closed issues
MMDetection: Community

From more than 10 institutes/companies
MMDetection: Model Definition

Two-stage Detector

- Sliding Window Anchors
- Dense Predictor
- RPN
- Proposals

Single-stage Detector

- Sliding Window Anchors
- Classification & Regression
# optimizer
optimizer = dict(type='SGD', lr=0.02, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=dict(max_norm=35, norm_type=2))

# learning policy
lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=1.0 / 3,
    step=[8, 11])
checkpoint_config = dict(interval=1)
log_config = dict(
    interval=50,
    hooks=[
        dict(type='TextLoggerHook'),
        dict(type='TensorboardLoggerHook')
    ])
Thank You