Instance segmentation
Semantic segmentation

Label every pixel, including the background (sky, grass, road)

Do not differentiate between the pixels coming from instances of the same class
Instance segmentation

Label every pixel, including the background (sky, grass, road)

Do not label pixels coming from uncountable objects (sky, grass, road)

Do not differentiate between the pixels coming from instances of the same class

Differentiate between the pixels coming from instances of the same class
Instance segmentation methods

Proposal-based

1. Proposals
2. Assign a class

FCN-based

1. Semantic segmentation
2. Find instances
Instance segmentation methods

Proposal-based
1. Proposals
2. Assign a class

vs.

FCN-based
1. Semantic segmentation
2. Find instances
FCN-based methods

A semantic map

We already know how to obtain this!
Why FCN-based?

- Fully Convolutional Networks for Semantic Segmentation
FCN-based methods

• X. Liang et al. “Proposal-free Network for Instance-level Object Segmentation“. Arxiv 2015

• A. Kirillov et al. „InstanceCut: from Edges to Instances with MultiCut“. CVPR 2017

• M. Bai and R. Urtasun “Deep Watershed Transform for Instance Segmentation“. CVPR 2017
Instances through clustering

A. Kirillov et al. „InstanceCut: from Edges to Instances with MultiCut“. CVPR 2017
Instance segmentation methods

Proposal-based

1. Proposals

2. Assign a class

FCN-based

1. Semantic segmentation

2. Find instances
Proposal-based methods

Bounding boxes....

We already know how to obtain those!
Proposal-based methods

• B. Hariharan et al. “Simultaneous Detection and Segmentation”. ECCV 2014

SDS

- SDS: Simultaneous Detection and Segmentation

MNC

- MNC: Multi-task network cascades
IS: the best of both worlds

Proposal-based

1. Proposals

2. Assign a class

(Proposals)

FCN-based

1. Semantic segmentation

2. Find instances

Person 1  Person 2  Person 3  Person 4  Person 5

Person 1  Person 2  Person 3  Person 4  Person 5
Mask R-CNN
What is Mask-RCNN?

- Starting from the Faster R-CNN architecture

Image → CNN → Region Proposal Network → Classification head → Bounding box regression head
What is Mask-RCNN?

- Faster R-CNN + FCN for segmentation
What is Mask-RCNN?

• Faster R-CNN + FCN for segmentation

Mask loss = binary cross entropy per pixel for the k semantic classes

He at al. “Mask R-CNN” ICCV 2017
Mask R-CNN

Most of features are shared

Object recognition head

Segmentation head

Faster R-CNN w/ ResNet [19]
Resolution vs. computation

output resolution is a tradeoff between computational cost and level of detail
Detection vs. segmentation

- Detection: for object classification, you require **invariant** representations

Translation invariance: wherever the penguin is in the image, I still want to have “penguin” as my classification output
Detection vs. segmentation

- Detection: for object classification, you require **invariant** representations
- Segmentation: you require **equivariant** representations
  - Translated object $\xrightarrow{\text{\相交}}$ Translated mask
  - Scaled object $\xrightarrow{\text{\相交}}$ scaled mask
  - For semantic segmentation, small objects are less important (less pixels), but for instance segmentation, all objects (no matter the size) are equally important
Mask-RCNN: operations

- What operations are equivariant?

Features extraction = convolutional layers equivariant

Segmentation head is a fully convolutional network equivariant
Mask-RCNN: operations

- What operations are equivariant?

Features extraction - convolutional layers equivariant

Segmentation head is a fully convolutional network equivariant

Fully connected layers and global pooling layers give invariance!
Recall: RoI pooling

- Region of Interest Pooling: for every proposal

Image → CNN → Feature map

We put a $H \times W$ grid on top

Zoom in

Feature map $(H \times w \times C)$
Recall: RoI pooling

• Let us look at sizes

Image 400x400

Box 300x150

CNN

Feature map (65x65xC)

Zoom in

Feature map (4x6xC)

We put a 4x6 grid on top

Box height

65*300/400=48.75

Quantization effect - chose 48

Not suitable to extract pixel-wise precise masks
Mask-RCNN: operations

- Make all operations equivariant

![Diagram showing the exchange of RoI pooling by an equivariant operation - RoI Align]

Fully connected layers and global pooling layers give invariance!

Exchange RoI pooling by an equivariant operation = RoI Align
RoIAlign

- Erase quantization effects

Image 400x400

CNN

Feature map (65x65xC)

Box height 65\times 300/400 = 48.75

Chose 48.75

Feature map (4x6xC)

Zoom in

We put a 4x6 grid on top

Box 300x150
To obtain the value use bilinear interpolation.

Feature map

Each unit is sampled 4 times.

Grid points for bilinear interpolation

Max pooling on the 4 positions to obtain one output value.

ROIAlign

(Variable size Roi)

RoIAlign output

(Fixed dimensional representation)
RoI Pooling: Recall

Model feature mapping process
Next, we're using one of the proposed RoIs (145x200 box) and try to map it onto the feature map. Because not all of our object dimensions can be divided by 32, we're placing RoI not align with our grid.

- (9.25, 6) - top left corner
- 6.25 - width
- 4.53 - height
RoI Pooling: Recall

Pooling layer

3x3 RoI Pooling
RoI Pooling: Recall

Quantization when mapping and pooling
Divide it into 9 boxes (because in our case the dimensions of our RoI Align are $3 \times 3$).
Divide it into 9 boxes (because in our case the dimensions of our RoI Align are $3 \times 3$). That gives us a box with a height of 1.51 and a width of 2.08
Get 4 sampling points. We get them by dividing the height and the width of the box by 3. For the first point:

- $X = X_{\text{box}} + (\text{width}/3) \cdot 1 = 9.94$
- $Y = Y_{\text{box}} + (\text{height}/3) \cdot 1 = 6.50$

where $(X_{\text{box}}, Y_{\text{box}}) = (9.25, 6)$.

For the second point:

- $X = X_{\text{box}} + (\text{width}/3) \cdot 1 = 9.94$
- $Y = Y_{\text{box}} + (\text{height}/3) \cdot 2 = 7.01$
RoI Align

Do bilinear interpolation following the equation:

$$P \approx \frac{y_2 - y}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{21} \right) + \frac{y - y_1}{y_2 - y_1} \left( \frac{x_2 - x}{x_2 - x_1} Q_{12} + \frac{x - x_1}{x_2 - x_1} Q_{22} \right)$$

Bilinear Interpolation equation
RoI Align

3x3 RoIAlign
RoI Align

Do this for every channel.
What is Mask-RCNN?

- Faster R-CNN + FCN for segmentation

Mask loss = binary cross entropy per pixel for the k semantic classes

He et al. “Mask R-CNN” ICCV 2017
Mask R-CNN: qualitative results
Mask R-CNN: qualitative results
Mask R-CNN: qualitative results
Mask R-CNN: qualitative results
Model a keypoint’s location as a one-hot mask, and adopt Mask R-CNN to predict $K$ masks, one for each of $K$ keypoint types (e.g., left shoulder, right elbow). This demonstrates the flexibility of Mask R-CNN.
Improving Mask-RCNN

- One problem with Mask R-CNN is that the mask quality score is computed as the confidence score for the bounding box.

Recall the mask loss just evaluates if the pixels have the correct semantic class, not the correct instance!

Both instances have the same class = person

The only way the “instance” is evaluated is through the box loss.
Mask IoU head

Measure the intersection over union between the predicted mask and ground truth mask
Typically, Mask scoring R-CNN gives lower confidence scores than Mask R-CNN, which corresponds to masks not being perfect (IoU < 1.0).

This tiny modification achieves SOTA results.
Is one-stage vs two-stage also applicable to masks?
One-stage vs two-stage detectors

Faster R-CNN

Slower, but has higher performance

YOLO

Faster, but has lower performance
One-stage vs two-stage instance segmenters

Mask R-CNN

Slower, but has higher performance

YOLACT

Faster, but has lower performance
YOLO with masks?

“Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them.”

– Joseph Redmon, YOLOv3
YolACT*

*You Only Look At Coefficients
YOLACT: idea

YOLACT: idea

1) Generate mask prototypes
YOLACT: idea

1) Generate mask prototypes

2) Generate mask coefficients
YOLACT: idea

1) Generate mask prototypes

2) Generate mask coefficients

3) Combine (1) and (2)
YOLACT: backbone

Feature Pyramid

Features computed in different scales

ResNet-101
YOLACT: protonet

Generate $k$ prototype masks. $k$ is not the number of classes, but is a hyperparameter.
YOLACT: protonet

- Fully convolutional network

Similar to the mask branch in Mask R-CNN.

However, no loss function is applied on this stage.
YOLACT: mask coefficients

Predict a coefficient for every predicted mask.
YOLACT: mask coefficients

The network is similar but shallower than RetinaNet

Predict one class per anchor box

Predict the regression per anchor box

Predict k coefficients (one per prototype mask) per anchor
YOLACT: mask assembly

1. Do a linear combination between the mask coefficients and the mask prototypes.
2. Predict the mask as $M = \sigma(PC^T)$ where $P$ is a $(HxWxK)$ matrix of prototype masks, $C$ is a $(NxK)$ matrix of mask coefficients surviving NMS, and $\sigma$ is a nonlinearity.
YOLACT: loss function

Cross-entropy between the assembled masks and the ground truth, in addition to the standard losses (regression for the bounding box, and classification for the class of the object/mask).
YOLACT: qualitative results
For large objects, the quality of the masks is even better than those of two-stage detectors
So, which segmenter to use?

![Graph showing performance comparison between different segmenters with YOLACT highlighted as the best choice.](Image)
YOLACT++: improvements

• A specially designed version of NMS, in order to make the procedure faster.

• An auxiliary semantic segmentation loss function performed on the final features of the FPN. The module is not used during the inference stage.

Panoptic segmentation
Panoptic segmentation

- Semantic segmentation
- Instance segmentation
Panoptic segmentation

Semantic segmentation

Instance segmentation

FCN-like + Mask R-CNN
Panoptic segmentation

Semantic segmentation + Instance segmentation = Panoptic segmentation

FCN-like + Mask R-CNN = UPSNet
Panoptic segmentation

It gives labels to uncountable objects called "stuff" (sky, road, etc), similar to FCN-like networks.

It differentiates between pixels coming from different instances of the same class (countable objects) called "things" (cars, pedestrians, etc).
Panoptic segmentation

Problem: some pixels might get classified as stuff from FCN network, while at the same time being classified as instances of some class from Mask R-CNN (conflicting results)!
Panoptic segmentation

Solution: Parametric-free panoptic head which combines the information from the FCN and Mask R-CNN, giving final predictions.

Xiong et al., “UPSNet: A Unified Panoptic Segmentation Network”. CVPR 2019
Network architecture

- **Shared features**
- **Separate heads**
- **Putting it together**
Network architecture

Shared features

Separate heads

Putting it together
The semantic head

As all semantic heads, fully convolutional network.

New: deformable convolutions!
Dilated (atrous) convolutions 2D

(a) the dilation parameter is 1, and each element produced by this filter has receptive field of $3 \times 3$.

(b) the dilation parameter is 2, and each element produced by it has receptive field of $5 \times 5$.

(c) the dilation parameter is 3, and each element produced by it has receptive field of $7 \times 7$. 
Deformable convolutions: generalization of dilated convolutions when you learn the offset
Deformable convolutions

Regular convolution
\[ y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n) \]

Deformable convolution
\[ y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \]

where \( \Delta p_n \) is generated by a sibling branch of regular convolution
Deformable convolutions

The deformable convolution will pick the values at different locations for convolutions conditioned on the input image of the feature maps.
The Panoptic head

- Mask logits from the instance head
- Object logits coming from the semantic head (e.g., car)
- Stuff logits coming from the semantic head (e.g., sky)
The Panoptic head

Mask logits from the instance head

Object logits coming from the semantic head (e.g., car)

Stuff logits coming from the semantic head (e.g., sky)

This can be evaluated directly

Objects need to be masked by the instance
The Panoptic head

Perform softmax over the panoptic logits. If the maximum value falls into the first stuff channels, then it belongs to one of the stuff classes. Otherwise the index of the maximum value tells us the instance ID the pixel belongs to.

Read the details on how to use the unknown class

Xiong et al., "UPSNet: A Unified Panoptic Segmentation Network". CVPR 2019
Panoptic quality

• As in detection, we have to “match ground truth and predictions. In this case we have segment matching.

• Segment is matched if IoU>0.5. No pixel can belong to two predicted segments.
Panoptic segmentation: qualitative
Panoptic segmentation: qualitative
Of course, it can be done with Transformers

- A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.
## DETR Panoptic Segmentation – results

<table>
<thead>
<tr>
<th>Model</th>
<th>Backbone</th>
<th>PQ</th>
<th>SQ</th>
<th>RQ</th>
<th>PQ(^{th})</th>
<th>SQ(^{th})</th>
<th>RQ(^{th})</th>
<th>PQ(^{st})</th>
<th>SQ(^{st})</th>
<th>RQ(^{st})</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PanopticFPN++</td>
<td>R50</td>
<td>42.4</td>
<td>79.3</td>
<td>51.6</td>
<td>49.2</td>
<td>82.4</td>
<td>58.8</td>
<td>32.3</td>
<td>74.8</td>
<td>40.6</td>
<td>37.7</td>
</tr>
<tr>
<td>UPSnet</td>
<td>R50</td>
<td>42.5</td>
<td>78.0</td>
<td>52.5</td>
<td>48.6</td>
<td>79.4</td>
<td>59.6</td>
<td>33.4</td>
<td>75.9</td>
<td>41.7</td>
<td>34.3</td>
</tr>
<tr>
<td>UPSnet-M</td>
<td>R50</td>
<td>43.0</td>
<td>79.1</td>
<td>52.8</td>
<td>48.9</td>
<td>79.7</td>
<td>59.7</td>
<td>34.1</td>
<td>78.2</td>
<td>42.3</td>
<td>34.3</td>
</tr>
<tr>
<td>PanopticFPN++</td>
<td>R101</td>
<td>44.1</td>
<td>79.5</td>
<td>53.3</td>
<td>51.0</td>
<td>83.2</td>
<td>60.6</td>
<td>33.6</td>
<td>74.0</td>
<td>42.1</td>
<td>39.7</td>
</tr>
<tr>
<td>DETR</td>
<td>R50</td>
<td>43.4</td>
<td>79.3</td>
<td>53.8</td>
<td>48.2</td>
<td>79.8</td>
<td>59.5</td>
<td>36.3</td>
<td>78.5</td>
<td>45.3</td>
<td>31.1</td>
</tr>
<tr>
<td>DETR-DC5</td>
<td>R50</td>
<td>44.6</td>
<td>79.8</td>
<td>55.0</td>
<td>49.4</td>
<td>80.5</td>
<td>60.6</td>
<td>37.3</td>
<td>78.7</td>
<td>46.5</td>
<td>31.9</td>
</tr>
<tr>
<td>DETR-R101</td>
<td>R101</td>
<td>45.1</td>
<td>79.9</td>
<td>55.5</td>
<td>50.5</td>
<td>80.9</td>
<td>61.7</td>
<td>37.0</td>
<td>78.5</td>
<td>46.0</td>
<td>33.0</td>
</tr>
</tbody>
</table>
DETR Panoptic Segmentation – results
Metrics
Panoptic quality

\[ PQ = \frac{\sum_{(p,g) \in TP} \text{IoU}(p, g)}{|TP|} \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \]

- SQ: Segmentation Quality = how close the predicted segments are to the ground truth segment (does not take into account bad predictions!)

TP = True positive, FN = False negative, FP = false positive
Panoptic quality

\[ PQ = \frac{\sum_{(p,g) \in TP} \text{IoU}(p, g)}{|TP|} \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \]

- RQ: Recognition Quality = just like for detection, we want to know if we are missing any instances (FN) or we are predicting more instances (FP).

TP = True positive, FN = False negative, FP = false positive
Instance segmentation
Object Instance Segmentation as Voting
Sliding Window Approach

• DPM, RCNN families
• Densely enumerate box proposals + classify
• Tremendously successful paradigm, very well engineered
• SOTA methods are still based on this paradigm
Generalized Hough Transform

Before DPM, RCNN dominance: detection-as-voting
Hough Voting

- Detect analytical shapes (e.g., lines) as peaks in the dual parametric space
- Each pixel casts a vote in this dual space
- Detect peaks and 'back-project' them to the image space
Example: Line Detection

- Each edge point in image space casts a vote
Example: Line Detection

- Each edge point in image space casts a vote
- The vote is in the form of a line that crosses the point

\[
b = -x_0m + y_0
\]
Example: Line Detection

• Accumulate votes from different points in (discretized) parameter space
• Read-out maxima (peaks) from the accumulator
Object Detection as Voting

• Idea: Objects are detected as consistent configurations of the observed parts (visual words)
Object Detection

• Training

Interest point detection
(SIFT, SURF)

Center point voting

Leibe et al., Robust Object Detection with Interleaved Categorization and Segmentation, IJCV'08
Object Detection

- Inference (test time)
Back to the future

• Back to 2020…

• We can use pixel consensus voting for panoptic segmentation (CVPR 20)
The instance voting branch predicts for every pixel whether the pixel is part of an instance mask, and if so, the relative location of the instance mask centroid.
In a Nutshell

1. Discretize regions around each pixel.
2. Every pixel votes for a centroid (or no centroid for “stuff”) over a set of grid cells.
In a Nutshell

3. Vote aggregation probabilities at each pixel are cast to accumulator space via (dilated) transposed convolutions
4. Detect objects as 'peaks' in the accumulator space
In a Nutshell

5. Back-projection of 'peaks' back to the image to get an instance masks

6. Category information provided by the parallel semantic segmentation head
Voting Lookup Table

- Discretize region around the pixel: $M \times M$ cells converted into $K=17$ indices.
Voting Lookup Table

- The vote should be cast to the center, which is the red pixel, which corresponds to position 16.
Voting

- At inference, instance voting branch provides tensor of size $[H, W, K+1]$
- Softly accumulate votes in the voting accumulator. **How?**

<table>
<thead>
<tr>
<th>10</th>
<th>10</th>
<th>10</th>
<th>9</th>
<th>9</th>
<th>9</th>
<th>16</th>
<th>16</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
</tbody>
</table>

Example: for the blue pixel, we get a vote for index 16 with 0.9 probability (softmax output)

- Transfer 0.9 to cell 16 -- (dilated) transposed convolution
- Evenly distribute among pixels, each gets 0.1 -- average pooling
Transposed Convolutions

• Take a single value in the input
• Multiply with a kernel and *distribute* in the output map
  • Kernel *defines* the amount of the input value that is being distributed to each of the output cells
• For the purpose of vote aggregation, however, we fix the kernel parameters to 1-hot across each channel that marks the target location.
Voting - Implementation

• Output tensor: \([H, W, K+1]\)
• Example: 9 inner, 8 outer bins, \(K=17\)
• Split the output tensor to two tensors: \([H, W, 9], [H, W, 8]\)
  • Apply two transposed convolutions, with kernel of size \([3, 3, 9],\) \(\text{stride}=1\) and \([3, 3, 8],\) \(\text{stride}=3\)
  • Pre-fixed kernel parameters; 1-hot across each channel that marks the target location
• Dilation \(\Rightarrow\) spread votes to the outer ring
• Smooth votes evenly via average pooling
Object Detection

- Peaks in the heatmap -- consensus detections
- Thresholding + connected components
Object Localization

- Vote back-projection
  - For every peak, determine pixels that favor this region above all others
Object Localization

- **Idea:** determine which pixels could have voted for a specific object center
  - Query filter
- **Examine votes**
  - Vote argmax
- **Find “consensus”**
  - Equality test

<table>
<thead>
<tr>
<th>Voting filter</th>
<th>Query filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 11 10 9 24</td>
<td>19 18 17 16</td>
</tr>
<tr>
<td>13 2 1 8 23</td>
<td>19 18 17 16</td>
</tr>
<tr>
<td>14 3 0 7 22</td>
<td>14 15 16 17</td>
</tr>
<tr>
<td>15 4 5 6 21</td>
<td>13 14 15 16</td>
</tr>
<tr>
<td>16 17 18 19 20</td>
<td>12 13 14 15</td>
</tr>
</tbody>
</table>

**Spatial Inversion**

- **Bottom-left pixel should have voted for ‘8’ if I’m the instance center!**
- **My center is at pixel 8!**
Fine-grained Scene Interpretation

- Individual objects, surfaces (things and stuff)
- Mobile robots
  - Reason about the drivability of surfaces.
  - The type of objects and obstacles.
  - The intent of other agents in the vicinity.
Instance segmentation