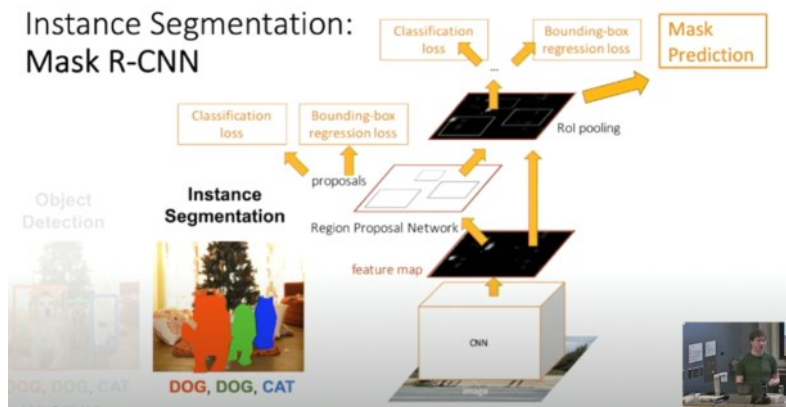


## 6 Segmentation

2024年12月18日 19:58

label each pixel in the image with a label. 不知道哪几个pixel属于cow, 有多少只cow

1. patching window. inefficient, not shared features between patches
2. fully conv: bunch of conv o make predictions for pixels all at once.
  - a. loss: per-pixel cross-entropy
  - b. all conv
    - i. effective receptive field size is linear.
    - ii. expensive computation
  - c. conv downsampling then **upsampling**
3. Things (individual obj instances), stuff (cannot separa)
  - a. obj detection: only things
  - b. segmentation: semantic:both; **instance segmentation** (both, obj detec then predict a seg)
4. Mask R-CNN



fix functions:

Unpooling

Bed of Nails

Nearest Neighbor

1	2
3	4

Input  
C x 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output  
C x 4 x 4

1	2
3	4

Input  
C x 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output  
C x 4 x 4

In-Network Upsampling: Bilinear Interpolation

1	2
3	4

Input: C x 2 x 2

1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{[x] - 1, \dots, [x] + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

$j \in \{[y] - 1, \dots, [y] + 1\}$

In-Network Upsampling: Bicubic Interpolation

1	2
3	4

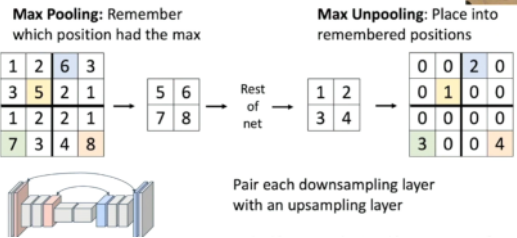
Input: C x 2 x 2

0.68	1.02	1.56	1.89
1.35	1.68	2.23	2.56
2.44	2.77	3.32	3.65
3.11	3.44	3.98	4.32

Output: C x 4 x 4

Use **three** closest neighbors in x and y to construct **cubic** approximations  
(This is how we normally resize images!)

In-Network Upsampling: “Max Unpooling”



和前一步的上采样结合

Learnable upsampling: transposed convolution