

分割

Segmentation

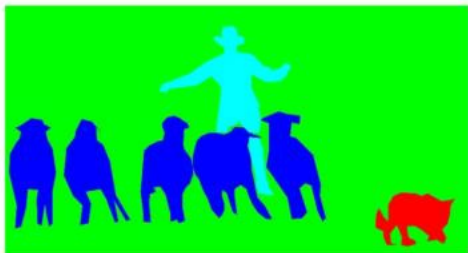
分割

- 介绍
- 发展
- 前沿

分割 Segmentation

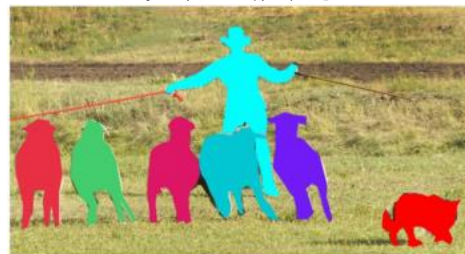


分类级别



Semantic segmentation

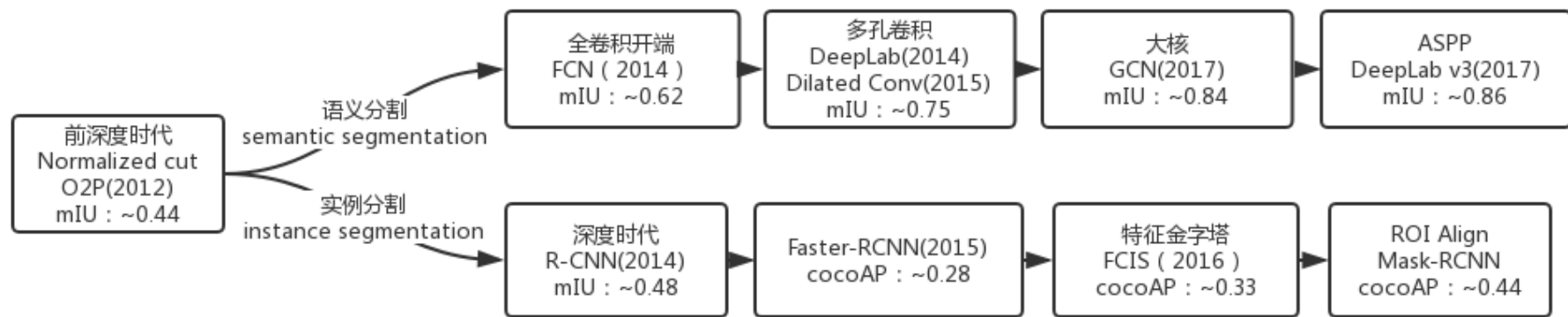
实例级别



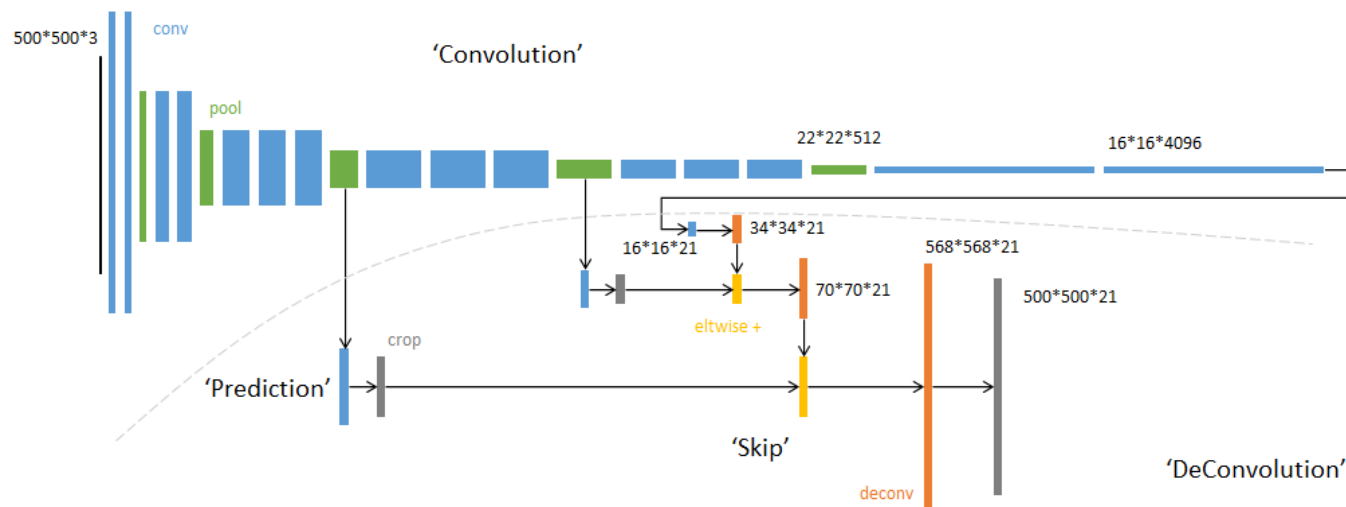
Instance segmentation



分割的发展

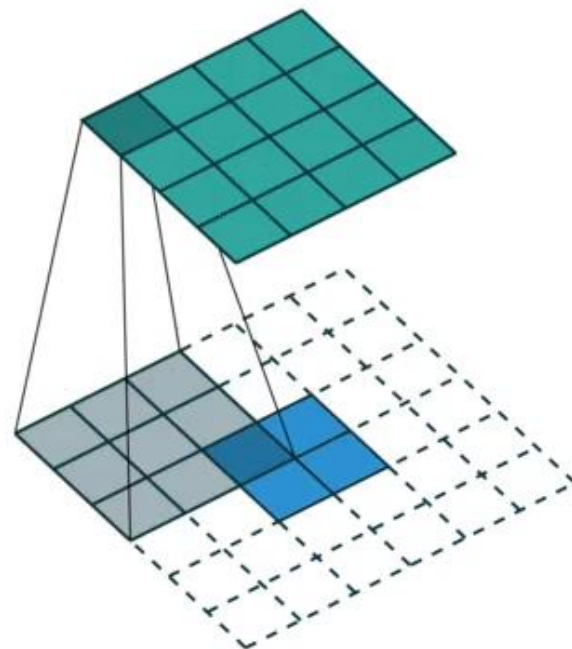


- Fully Convolutional Networks for Semantic Segmentation
arXiv:1411.4038
- 全卷积
- 多层feature跳接结构 (Skip Architecture)

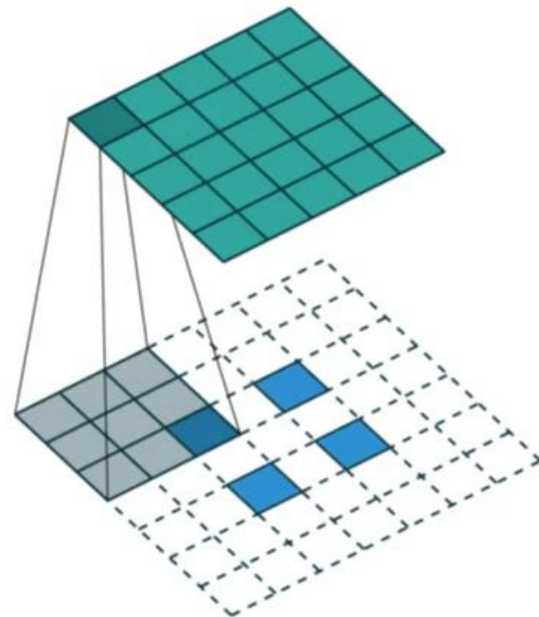
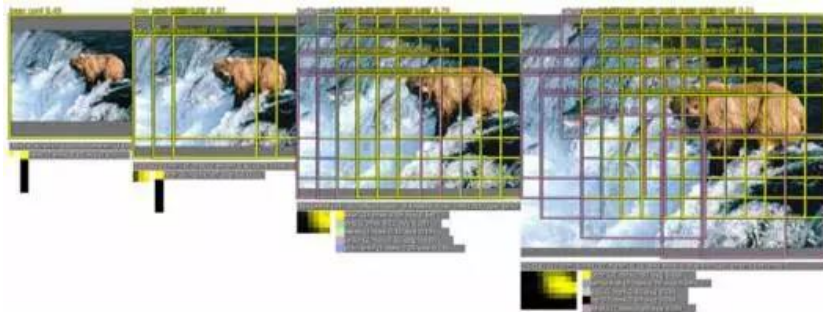


- Fully Convolutional Networks for Semantic Segmentation
arXiv:1411.4038
- 全卷积
- 多层feature跳接结构 (Skip Architecture)

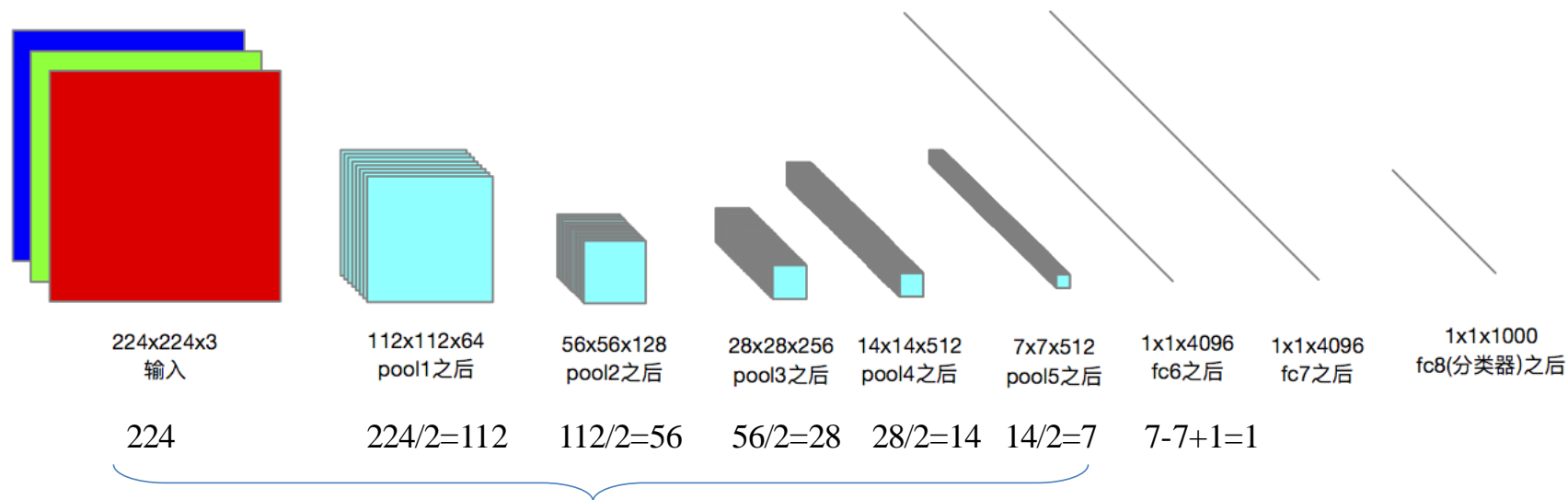
0.0625,	0.1875,	0.1875,	0.0625
0.1875,	0.5625,	0.5625,	0.1875
0.1875,	0.5625,	0.5625,	0.1875
0.0625,	0.1875,	0.1875,	0.0625



- Fully Convolutional Networks for Semantic Segmentation
arXiv:1411.4038
- 全卷积
- 多层feature跳接结构 (Skip Architecture)

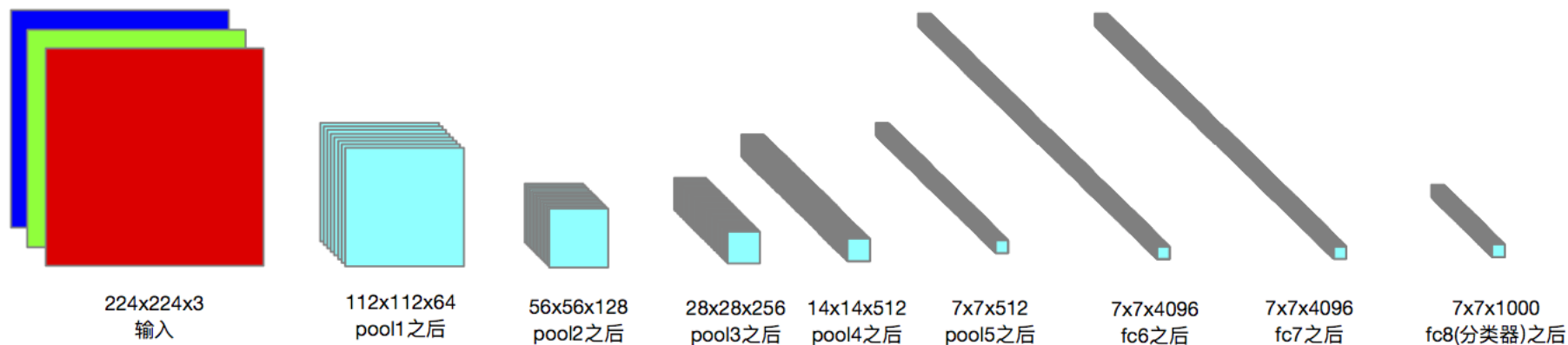


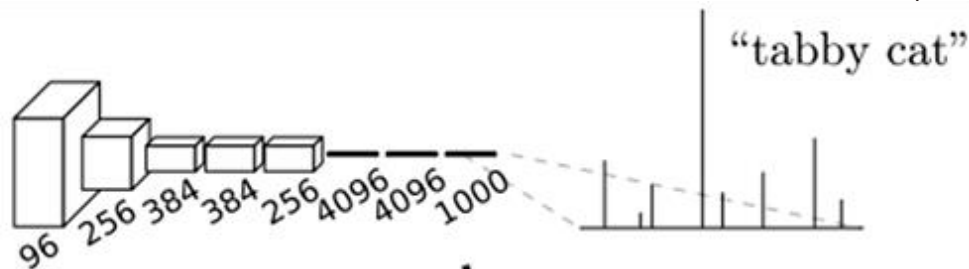
- 回忆一下经典的VGG-16



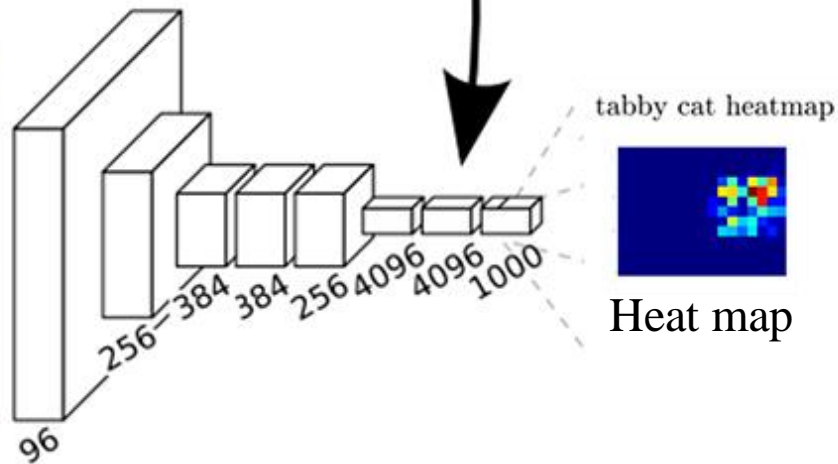
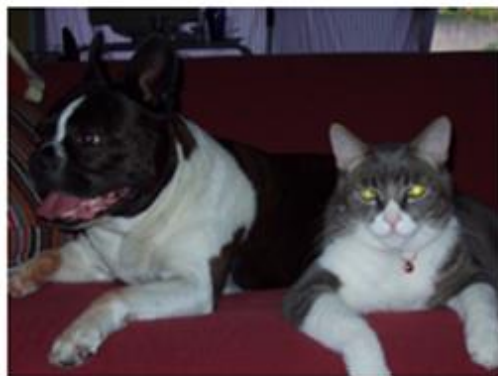
空间 (spatial) 尺度缩小了 $2^6=32$ 倍

- 将最后的7x7 conv (fc6) padding改为SAME





convolutionalization

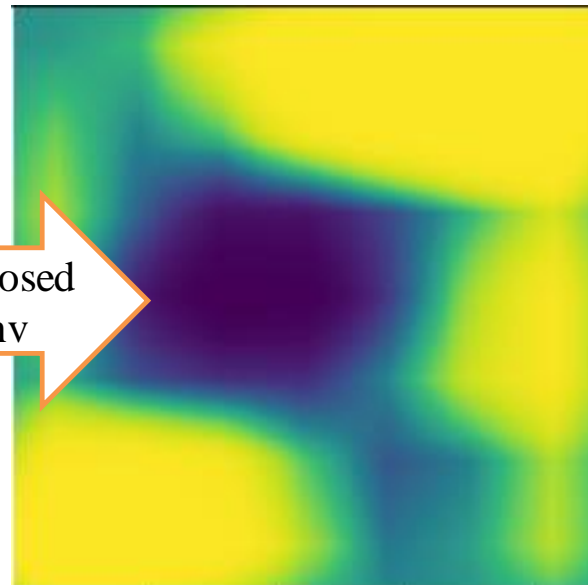




Heat map

```
0.00,0.00,0.00,0.01,0.01,0.01,0.01,0.02,0.02,0.02,0.02,0.03,0.03,0.03,0.03,0.03,0.03,0.02,0.02,0.02,0.02,0.01,0.01,0.01,0.01,0.00,0.00,
0.00,0.01,0.01,0.02,0.03,0.04,0.04,0.05,0.05,0.06,0.07,0.07,0.08,0.09,0.09,0.09,0.08,0.07,0.06,0.05,0.04,0.04,0.03,0.02,0.01,0.01,0.00,
0.00,0.01,0.02,0.04,0.05,0.06,0.07,0.08,0.09,0.10,0.11,0.12,0.13,0.14,0.15,0.15,0.16,0.13,0.11,0.10,0.09,0.08,0.07,0.06,0.05,0.04,0.03,0.02,0.01,0.00,
0.01,0.02,0.03,0.05,0.06,0.08,0.09,0.10,0.12,0.13,0.14,0.16,0.17,0.18,0.20,0.21,0.21,0.20,0.18,0.17,0.16,0.15,0.13,0.12,0.10,0.09,0.08,0.06,0.05,0.03,0.02,0.01,
0.01,0.02,0.04,0.06,0.08,0.10,0.11,0.12,0.13,0.17,0.18,0.20,0.22,0.24,0.25,0.27,0.27,0.27,0.26,0.24,0.22,0.20,0.18,0.17,0.13,0.11,0.10,0.09,0.06,0.05,0.03,0.01,
0.01,0.03,0.05,0.09,0.10,0.12,0.14,0.16,0.20,0.23,0.25,0.27,0.29,0.31,0.33,0.33,0.33,0.30,0.27,0.25,0.23,0.20,0.18,0.16,0.14,0.12,0.10,0.09,0.06,0.05,0.03,0.01,
0.01,0.04,0.06,0.09,0.11,0.14,0.17,0.19,0.22,0.24,0.27,0.29,0.32,0.34,0.37,0.39,0.39,0.37,0.34,0.32,0.29,0.27,0.24,0.22,0.20,0.17,0.14,0.11,0.09,0.06,0.04,0.01,
0.01,0.04,0.07,0.10,0.13,0.16,0.19,0.21,0.25,0.28,0.31,0.34,0.37,0.40,0.43,0.45,0.45,0.42,0.40,0.37,0.34,0.31,0.28,0.25,0.22,0.19,0.16,0.13,0.10,0.07,0.04,0.01,
0.02,0.05,0.08,0.12,0.15,0.18,0.22,0.25,0.28,0.32,0.35,0.38,0.42,0.45,0.48,0.51,0.51,0.48,0.45,0.42,0.38,0.35,0.32,0.28,0.25,0.22,0.18,0.15,0.12,0.09,0.06,0.02,
0.02,0.06,0.09,0.13,0.17,0.20,0.24,0.28,0.32,0.35,0.39,0.43,0.46,0.48,0.54,0.58,0.58,0.54,0.58,0.48,0.43,0.39,0.35,0.32,0.28,0.24,0.20,0.17,0.13,0.09,0.06,0.02,
0.02,0.06,0.10,0.14,0.18,0.23,0.27,0.31,0.35,0.39,0.43,0.47,0.51,0.55,0.59,0.64,0.64,0.59,0.55,0.51,0.47,0.43,0.39,0.35,0.31,0.27,0.23,0.18,0.14,0.10,0.06,0.02,
0.02,0.07,0.11,0.16,0.20,0.25,0.29,0.34,0.38,0.43,0.47,0.52,0.56,0.61,0.65,0.70,0.70,0.65,0.61,0.56,0.52,0.47,0.43,0.38,0.34,0.29,0.25,0.20,0.16,0.11,0.07,0.02,
0.02,0.07,0.12,0.17,0.22,0.27,0.32,0.37,0.42,0.46,0.51,0.56,0.61,0.66,0.71,0.76,0.76,0.71,0.66,0.61,0.56,0.51,0.46,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.08,0.13,0.18,0.24,0.29,0.34,0.40,0.45,0.50,0.55,0.61,0.66,0.71,0.76,0.82,0.82,0.76,0.71,0.66,0.61,0.56,0.51,0.46,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.08,0.14,0.20,0.25,0.31,0.37,0.42,0.48,0.54,0.59,0.65,0.71,0.76,0.82,0.88,0.88,0.82,0.76,0.71,0.66,0.61,0.56,0.51,0.46,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.09,0.15,0.21,0.27,0.33,0.39,0.45,0.51,0.58,0.64,0.70,0.76,0.82,0.88,0.94,0.94,0.88,0.82,0.76,0.70,0.64,0.58,0.51,0.45,0.39,0.33,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.09,0.16,0.22,0.28,0.34,0.40,0.46,0.52,0.58,0.64,0.70,0.76,0.82,0.88,0.94,0.94,0.88,0.82,0.76,0.70,0.64,0.58,0.51,0.45,0.39,0.33,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.09,0.17,0.23,0.29,0.35,0.41,0.47,0.53,0.59,0.65,0.71,0.76,0.82,0.88,0.94,0.94,0.88,0.82,0.76,0.71,0.65,0.59,0.54,0.48,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.03,0.09,0.18,0.24,0.29,0.34,0.40,0.45,0.50,0.55,0.61,0.66,0.71,0.76,0.82,0.82,0.76,0.71,0.66,0.61,0.56,0.51,0.46,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.02,0.07,0.12,0.17,0.22,0.27,0.32,0.37,0.42,0.46,0.51,0.56,0.61,0.66,0.71,0.76,0.76,0.71,0.66,0.61,0.56,0.51,0.46,0.42,0.37,0.32,0.27,0.22,0.17,0.12,0.07,0.02,
0.02,0.07,0.11,0.16,0.20,0.25,0.29,0.34,0.38,0.43,0.47,0.52,0.56,0.61,0.65,0.70,0.76,0.65,0.61,0.56,0.52,0.47,0.43,0.38,0.34,0.29,0.25,0.20,0.16,0.11,0.07,0.02,
0.02,0.06,0.10,0.14,0.18,0.23,0.27,0.31,0.35,0.39,0.43,0.47,0.51,0.55,0.59,0.64,0.64,0.59,0.55,0.51,0.47,0.43,0.39,0.35,0.31,0.27,0.23,0.18,0.14,0.10,0.06,0.02,
0.02,0.06,0.09,0.13,0.17,0.20,0.24,0.28,0.32,0.35,0.39,0.43,0.46,0.50,0.54,0.58,0.58,0.54,0.58,0.48,0.43,0.39,0.35,0.32,0.28,0.24,0.20,0.17,0.13,0.09,0.06,0.02,
0.02,0.05,0.08,0.12,0.15,0.18,0.22,0.25,0.28,0.32,0.35,0.38,0.42,0.45,0.48,0.51,0.51,0.48,0.45,0.42,0.38,0.35,0.32,0.28,0.25,0.22,0.18,0.15,0.12,0.09,0.05,0.02,
0.01,0.04,0.07,0.10,0.13,0.16,0.19,0.22,0.25,0.28,0.31,0.34,0.37,0.40,0.43,0.45,0.45,0.42,0.37,0.34,0.31,0.28,0.25,0.22,0.18,0.16,0.13,0.10,0.07,0.04,0.01,
0.01,0.04,0.06,0.09,0.11,0.14,0.17,0.19,0.22,0.24,0.27,0.29,0.32,0.34,0.37,0.39,0.39,0.37,0.34,0.32,0.29,0.27,0.24,0.22,0.19,0.16,0.13,0.10,0.07,0.04,0.01,
0.01,0.03,0.05,0.08,0.10,0.12,0.14,0.16,0.18,0.20,0.23,0.25,0.27,0.29,0.31,0.33,0.33,0.31,0.29,0.27,0.25,0.23,0.20,0.18,0.16,0.14,0.12,0.10,0.08,0.06,0.04,0.03,0.01,
0.01,0.03,0.04,0.06,0.08,0.10,0.11,0.13,0.15,0.17,0.18,0.20,0.22,0.24,0.25,0.27,0.27,0.26,0.24,0.22,0.20,0.18,0.17,0.13,0.11,0.10,0.09,0.06,0.05,0.04,0.03,0.01,
0.01,0.03,0.03,0.05,0.06,0.08,0.10,0.12,0.13,0.14,0.16,0.17,0.18,0.20,0.21,0.21,0.20,0.18,0.17,0.16,0.14,0.13,0.12,0.10,0.09,0.08,0.06,0.05,0.04,0.03,0.02,0.01,
0.00,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.10,0.11,0.12,0.13,0.14,0.15,0.15,0.14,0.13,0.12,0.11,0.10,0.09,0.08,0.07,0.06,0.05,0.04,0.03,0.02,0.01,0.01,0.00,
0.00,0.01,0.01,0.02,0.03,0.04,0.04,0.05,0.06,0.07,0.07,0.08,0.08,0.09,0.09,0.08,0.07,0.07,0.06,0.06,0.05,0.04,0.04,0.03,0.03,0.02,0.01,0.01,0.00,
0.00,0.00,0.00,0.01,0.01,0.01,0.01,0.02,0.02,0.02,0.02,0.03,0.03,0.03,0.03,0.03,0.02,0.02,0.02,0.02,0.01,0.01,0.01,0.01,0.00,0.00,0.00]
```

Tranposed
conv

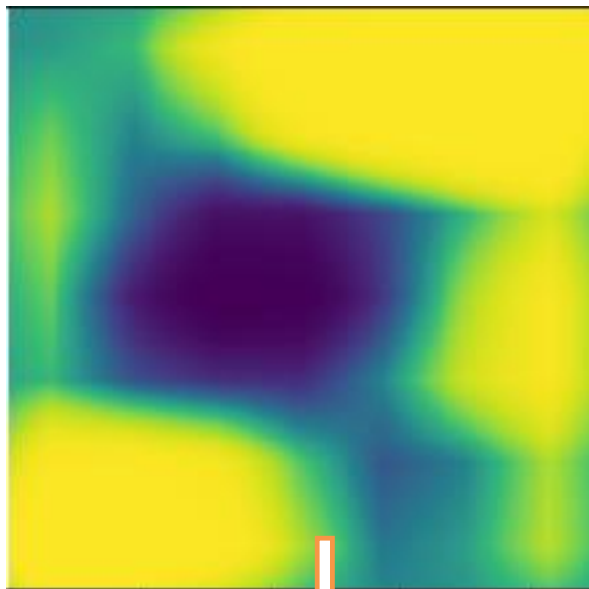


Bilinear interpolation



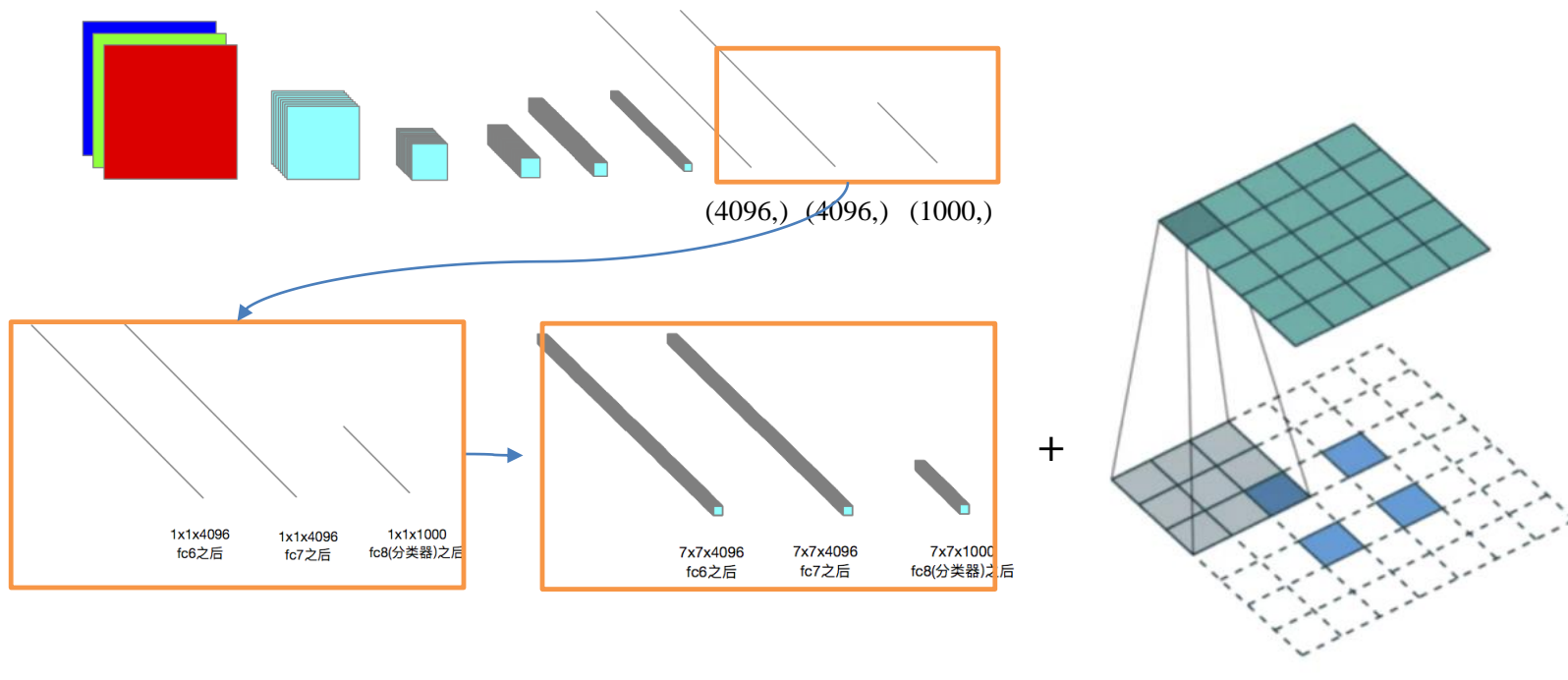
► FCN-CRF (conditional random fields)

$$P(\mathbf{X} = \mathbf{x}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp(-E(\mathbf{x}|\mathbf{I}))$$



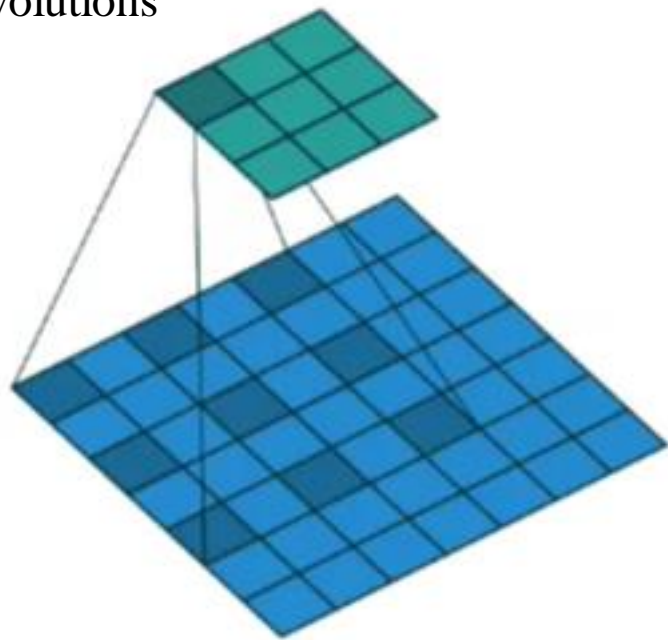
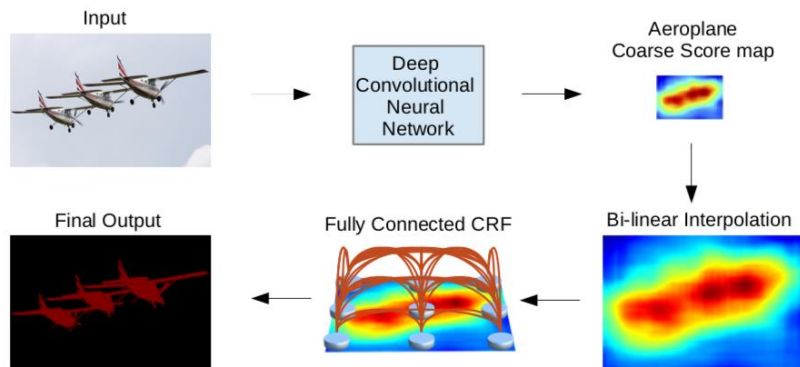
$$E(\mathbf{x}) = \sum_i \Psi_u(x_i) + \sum_{i < j} \Psi_p(x_i, x_j)$$





$$+ P(\mathbf{X} = \mathbf{x}|\mathbf{I}) = \frac{1}{Z(\mathbf{I})} \exp(-E(\mathbf{x}|\mathbf{I}))$$

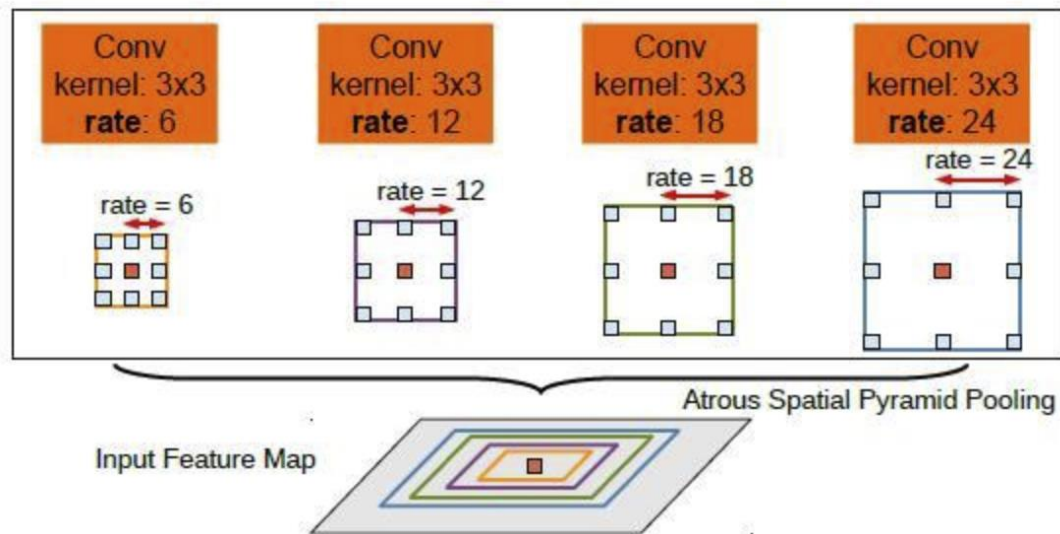
- Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs
arXiv:1412.7062
- Multi-Scale Context Aggregation by Dilated Convolutions
arXiv:1511.01722
- 利用多孔卷积替代掉原有VGG的Conv



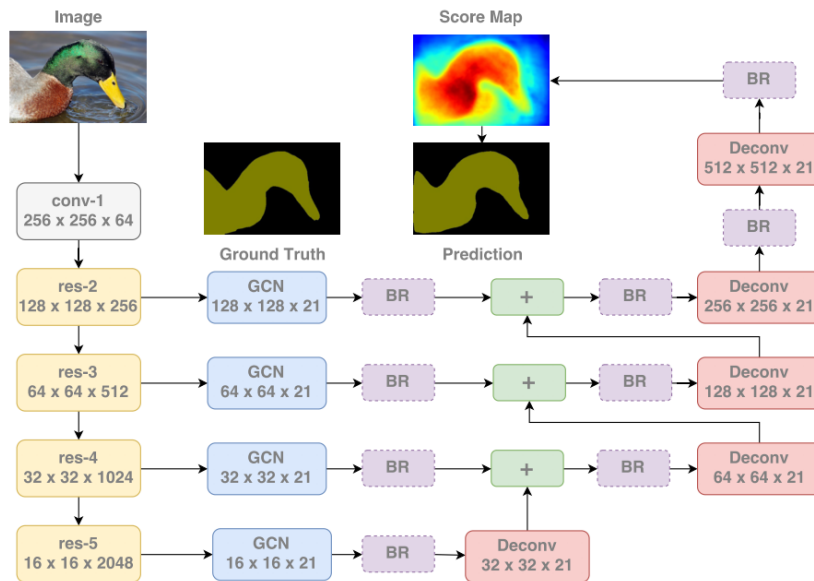


```
net = slim.max_pool2d(net, [2, 2], scope='pool3')
net = slim.repeat(net, 3, slim.conv2d, 512, [3, 3], scope='conv4')
net = slim.max_pool2d(net, [1, 1], stride=1, scope='pool4')
with tf.variable_scope('conv5'):
    with tf.variable_scope('conv5_1'):
        kernel = tf.Variable(tf.truncated_normal(shape=[3, 3, 512, 512]),
                              name='weights')
        biases = tf.Variable(tf.zeros([512]), name='biases')
        net = tf.nn.atrous_conv2d(net, kernel,
                                   rate=2, padding='SAME', name='conv2d')
        net = tf.nn.relu(net+biases)
```

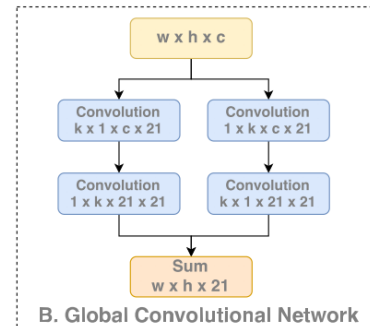
- DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
arXiv:1606.00915
- atrous spatial pyramid pooling(ASPP)



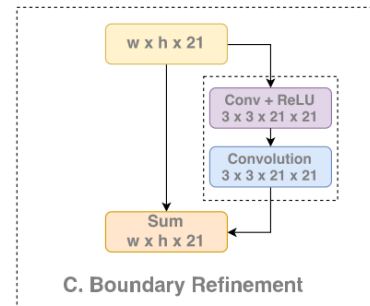
- Large Kernel Matters
 - Improve Semantic Segmentation by Global Convolutional Network
- arXiv:1703.02719
- 大内核 (Large kernel)



A. Whole Pipeline

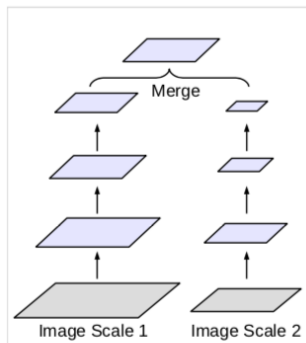


B. Global Convolutional Network

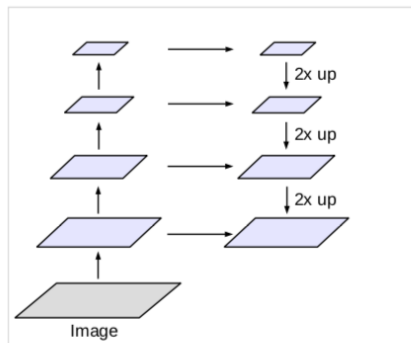


C. Boundary Refinement

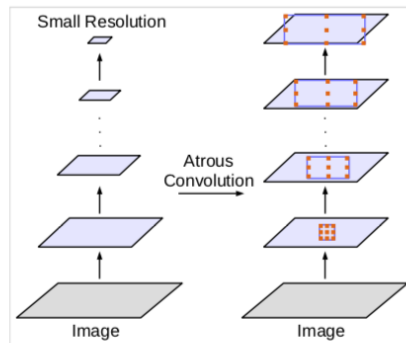
- Rethinking Atrous Convolution for Semantic Image Segmentation
arXiv:1706.05587
- 多尺度信息结合方式的探究
- 工程上的胜利



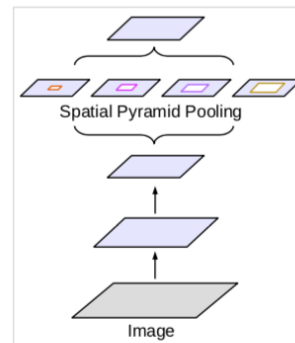
(a) Image Pyramid



(b) Encoder-Decoder

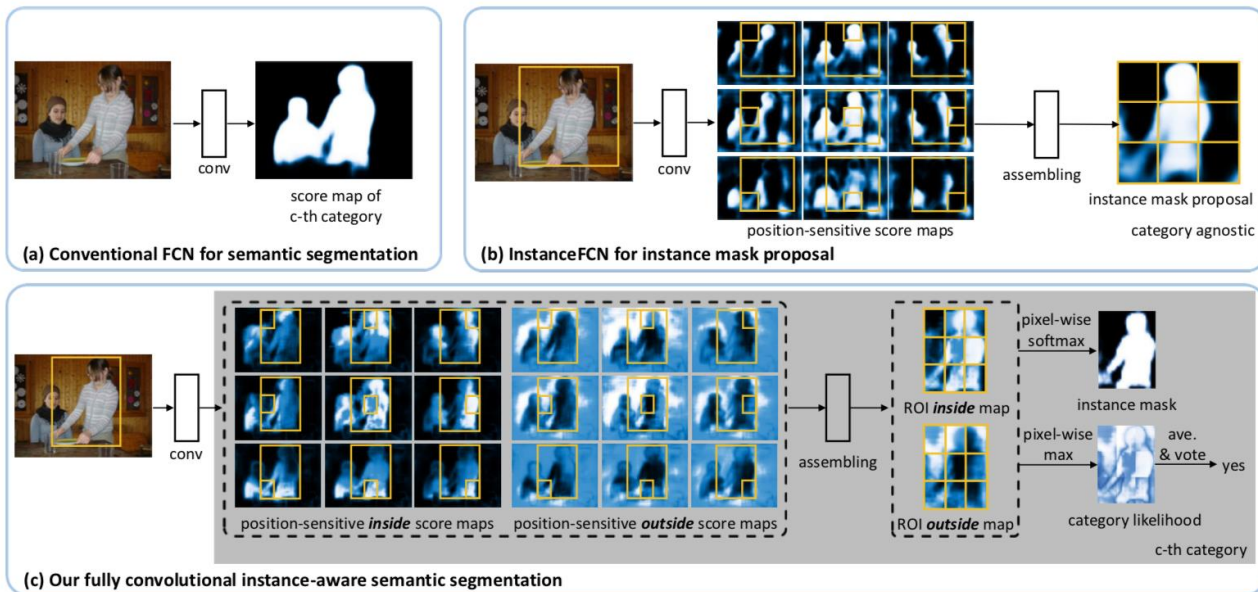


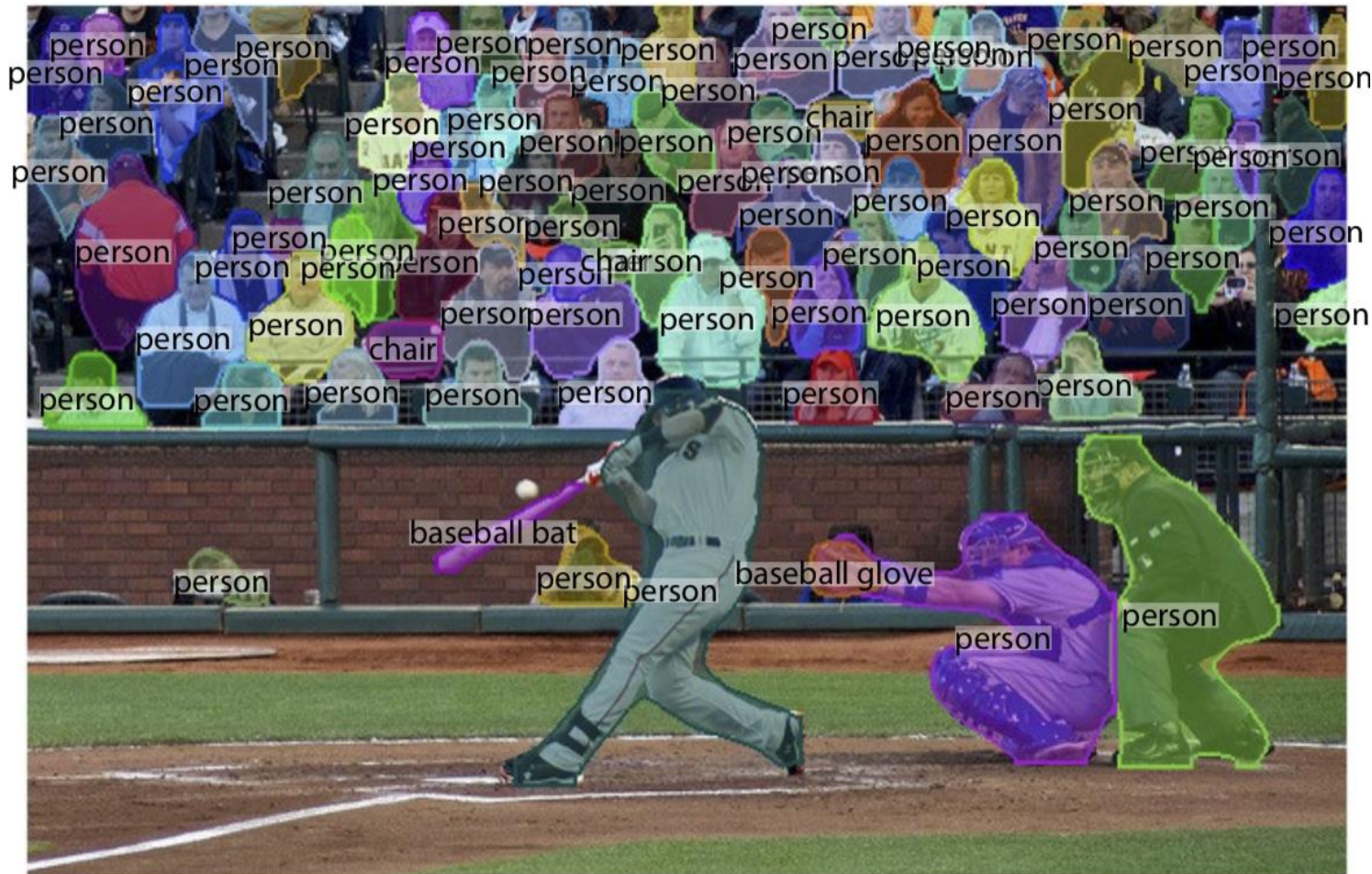
(c) Deeper w. Atrous Convolution



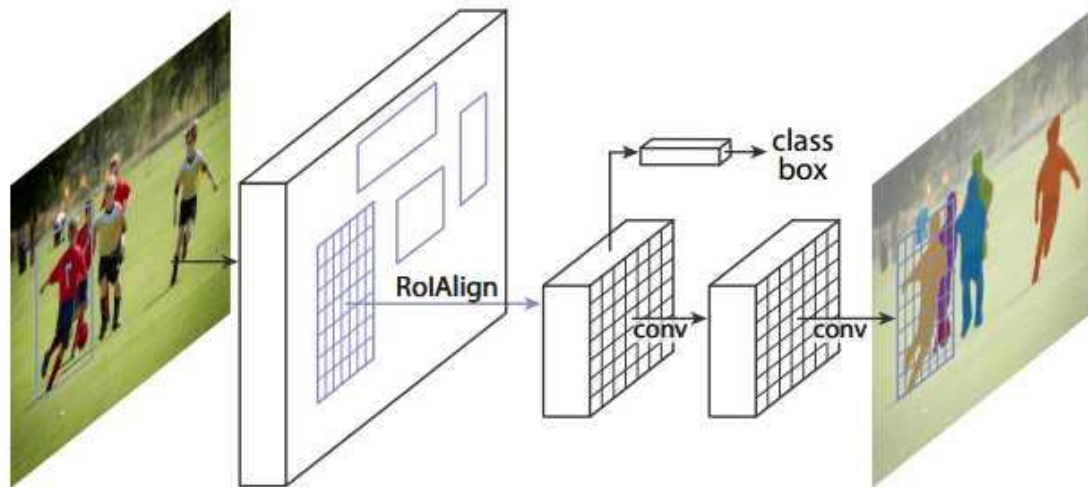
(d) Spatial Pyramid Pooling

- Fully Convolutional Instance-aware Semantic Segmentation
arXiv:1611.07709
- Inside/outside score



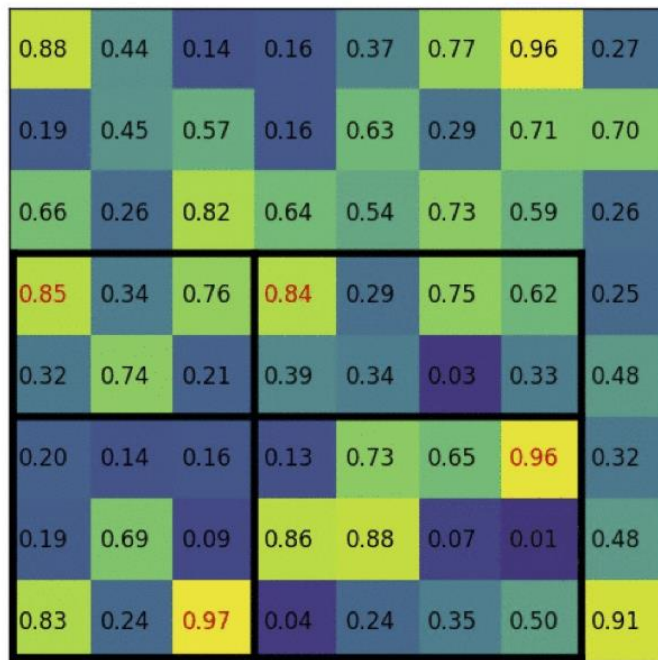
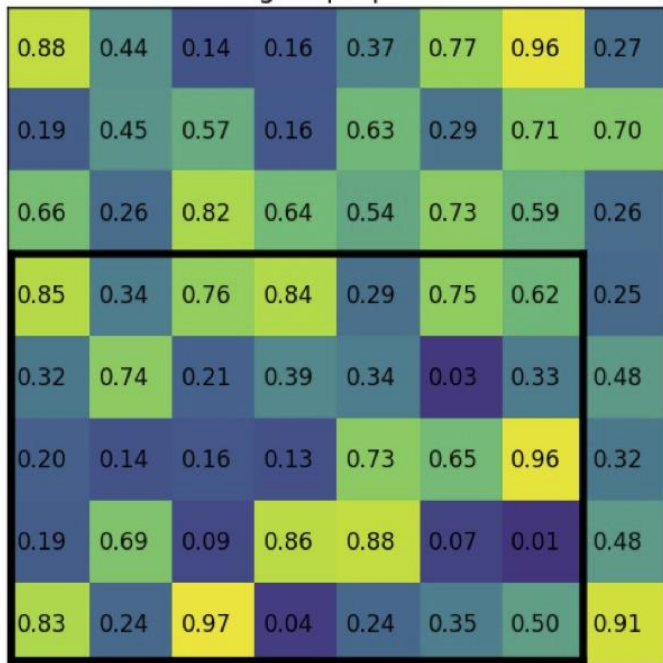


- Mask R-CNN
arXiv: 1703.06870
- 结构上：Faster R-CNN+FCN
- 为了得到更精确的mask位置，加入RoI Align





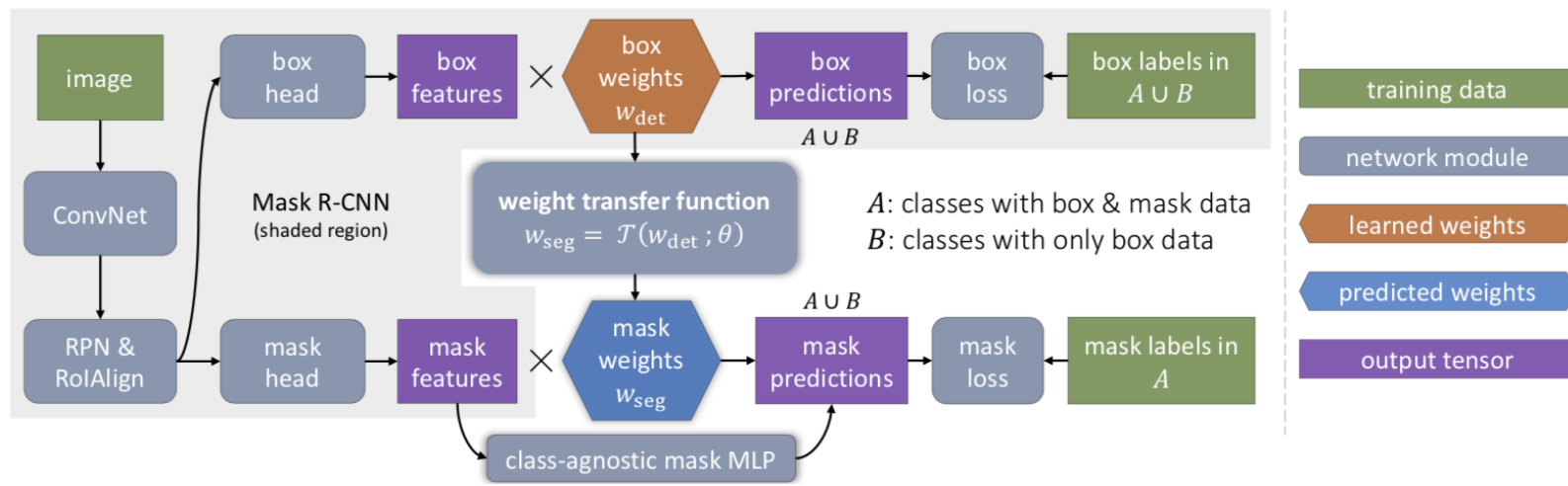
- 回顾一下ROI pooling



- ROI Align：采用双线性插值的方式，允许非整数像素的切割点



- Learning to segment every thing
arXiv: 1711.10370
- 采用迁移学习思路，学习一个映射
- 可能大幅度减小训练数据的成本



THANK YOU



AI100