Learning Deconvolution Network for Semantic Segmentation

최희정

0 Abstract

- DeconvNet은 새로운 semantic segmentation algorithm으로, VGG16 convolutional network와 deconvolution과 unpooling layers로 이루어진 deconvolution network로 구성되어 있다.

- Input image에서 뽑아낸 여러 proposal을 DeconvNet에 넣고, 결과로 나온 여러 semantic segmentation map들을 간단한 방식으로 계산해 최종 semantic segmentation map을 생성하는 모델이다.

- Deep deconvolution network와 proposal-wise 예측을 이용해 기존 FCN의 한계를 완화했으며,
  그 결과 DeconvNet은 detailed structure를 인식할 수 있고 multiple-scale의 object를 다룰 수 있다.

- PASCAL VOC 2012 dataset으로 FCN과 DeconvNet의 ensemble 모델이 best accuracy(72.5%)를 달성했다.

1 Introduction

- FCN은 기존의 CNN을 pixel-wise labeling 문제를 푸는 모델로 변경시킨 것으로,
  단순한 interpolation과 같은 deconvolution을 통해 coarse label map을 생성한다.

- FCN은 선택적으로 CRF(Conditional Random Field)를 output map에 적용하기도 한다.

- FCN의 장점은 whole image를 input으로 받아들여 빠르고 정확한 segmentation을 하는 것이다.

1 Introduction

- 하지만, FCN은 두 가지 한계점을 가진다.

1) fixed-size receptive field를 가지기 때문에 image에서 single-scale semantics만 다룰 수 있다.
즉, large object는 inconsistent label을 가지고,
small object는 배경으로 misclassified되는 경우가 발생할 수 있다. (skip layer을 도입해 이러한 한계점을 극복하려 했지만 여전히 한계가 존재함)

2) deconvolutional layer의 input인 label map이 coarse하고, deconvolution이 너무 단순하기 때문에 object의 detailed structure가 사라지거나 smooth해진다.
(DeconvNet은 이러한 한계를 CRF를 이용해 극복함)

(a) Inconsistent labels due to large object size

(b) Missing labels due to small object size

Figure 1. Limitations of semantic segmentation algorithms based on fully convolutional network. (Left) original image. (Center) ground-truth annotation. (Right) segmentations by [17]

1 Introduction

- 이러한 FCN의 한계를 극복하기 위한 Deconvnet의 main contribution은 다음과 같다.

  1) deconvolution, unpooling, ReLU layers로 이루어진 multi-layer deconvolution network를 학습해 detailed structure 한계를 극복할 수 있다.

  2) 여러 object proposal에 trained Deconvnet을 적용해 여러 개의 instance-wise segmentation을 얻고 마지막에 이것들을 종합해 최종 output을 내는 방식으로 진행되기 때문에 single-scale 한계를 극복하고, 더불어 detail을 더 잘 잡아낼 수 있다.

  3) 외부 데이터 없이 오직 PASCAL VOC 2012 dataset만으로 train했고 상호보완적인 특징을 가진 FCN과의 ensemble을 통해 best accuracy를 얻었다.

3 System Architecture

3.1. Architecture

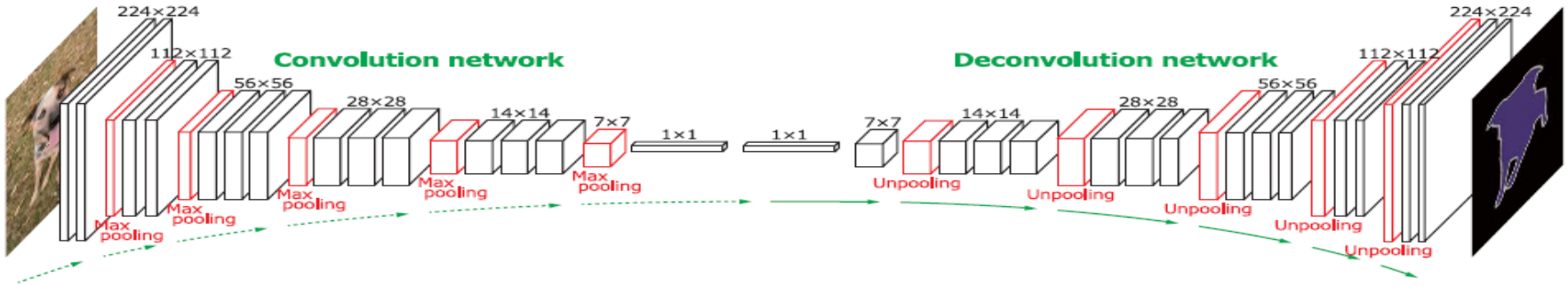


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

1) Convolution network

- input image를 multidimensional feature로 표현하는 feature extractor 역할이다.
- 마지막 classification layer를 제거한 VGG16 이용한다.
- 13개의 convolutional layer사이에 ReLU와 pooling layer를 넣고 마지막에 2-fully connected layer를 넣어 class-specific projection을 생성하는 구조이다.

3 System Architecture

3.1. Architecture

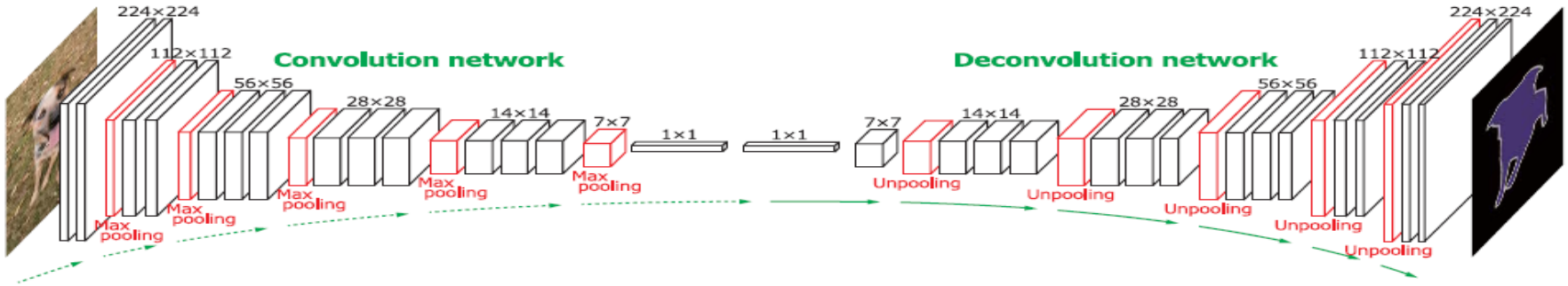


Figure 2. Overall architecture of the proposed network. On top of the convolution network based on VGG 16-layer net, we put a multi-layer deconvolution network to generate the accurate segmentation map of an input proposal. Given a feature representation obtained from the convolution network, dense pixel-wise class prediction map is constructed through multiple series of unpooling, deconvolution and rectification operations.

2) Deconvolution network

- Convolution network의 output으로 object segmentation을 생성하는 shape generator 역할이다.
- Convolution network의 mirrored version으로 deconvolution, unpooling, ReLU layers로 구성되어 있다.
- deconvolution과 unpooling operation으로 activation maps를 크게 만든다.

3 System Architecture

3.2. Deconvolution Network for Segmentation

3.2.1 Unpooling

- Convolution network에서 lower layer에 있는 pooling layer는 receptive field에 해당하는 activation을 하나의 대표 값으로 바꿔 noisy를 제거하는 역할을 하며, 이로 인해 upper layer에는 robust한 activation만 남게 되므로 classification이 잘 되게 한다.

- 하지만, pooling은 receptive field 안에 있는 공간정보를 없애므로, 이것은 semantic segmentation에 중요한 localization에 부정적인 영향을 미친다.

- 따라서, 이러한 문제를 해결하기 위해, DeconvNet은 pooling layer와 반대 operation을 하고 activation을 original size로 재구성하는 unpooling layer를 도입한다.

3 System Architecture

3.2. Deconvolution Network for Segmentation

3.2.1 Unpooling

- Convolution network에서 pooling이 일어날 때, max 값의 위치를 switch variable에 저장하고, 이 정보를 unpooling layer에서 위치 정보로 사용한다.

- 즉, FCN처럼 interpolation 하지 않고 max값을 receptive field의 크기로 확대시킬 때, Convolution network에서 max 값이 있던 위치에 그 값을 넣고 나머지는 0을 넣어 확대한다.

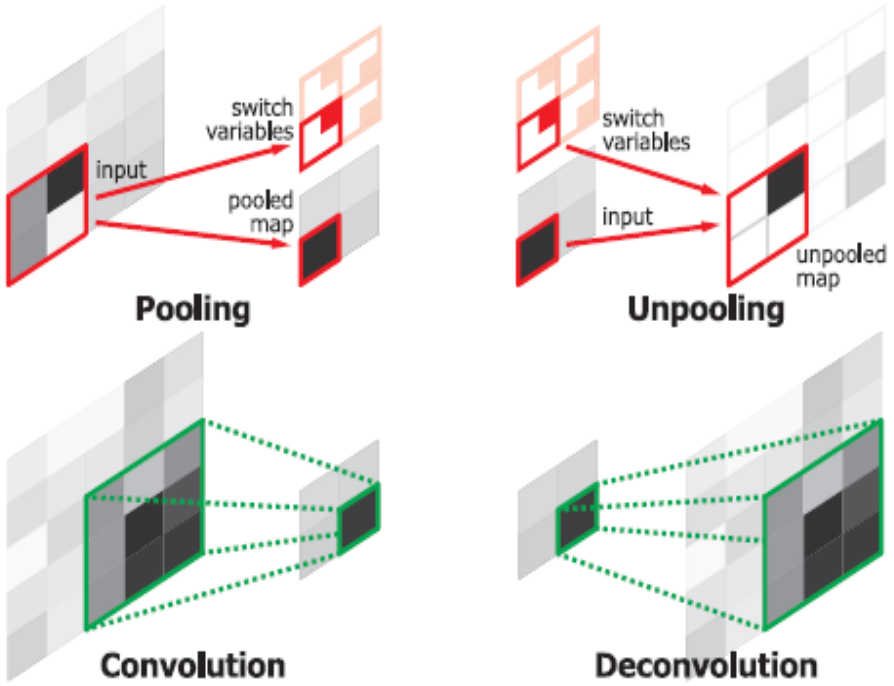


Figure 3. Illustration of deconvolution and unpooling operations.

3 System Architecture

3.2. Deconvolution Network for Segmentation

3.2.2 Deconvolution

- Unpooling layer의 output은 크기는 크지만 sparse하므로 deconvolution layer를 이용해 dense하게 만들어 준다.

- Convolution network의 convolutional layer와 같이 학습된 filter와의 연산을 통해 dense하게 만들며, convolutional layer와 다르게 하나의 input이 여러 output과 이어져있는 형태이다.

- Deconvolution layer의 output의 boundary를 crop해 이전 Unpooling layer의 output의 size와 맞춘다.

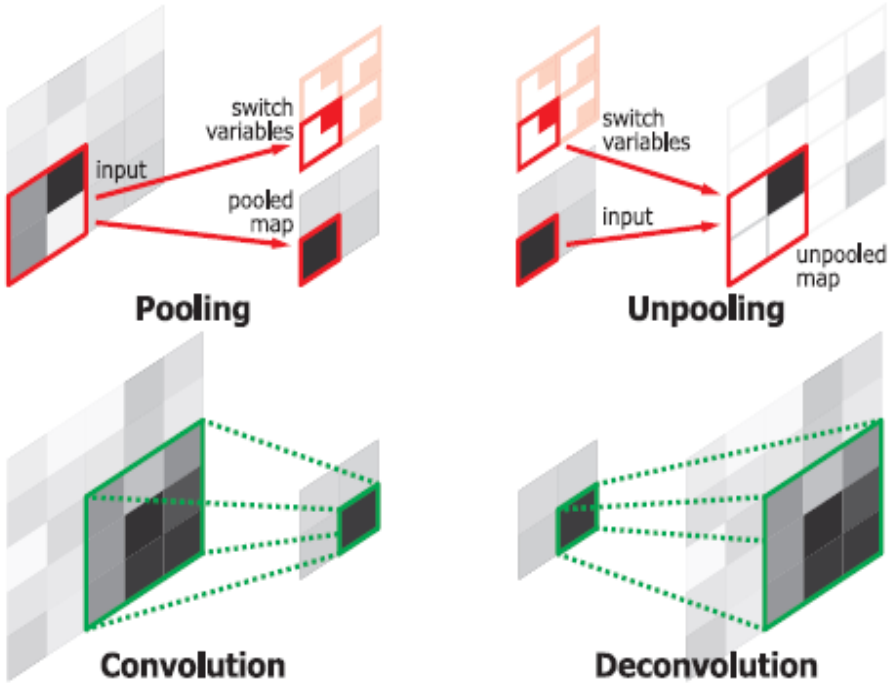
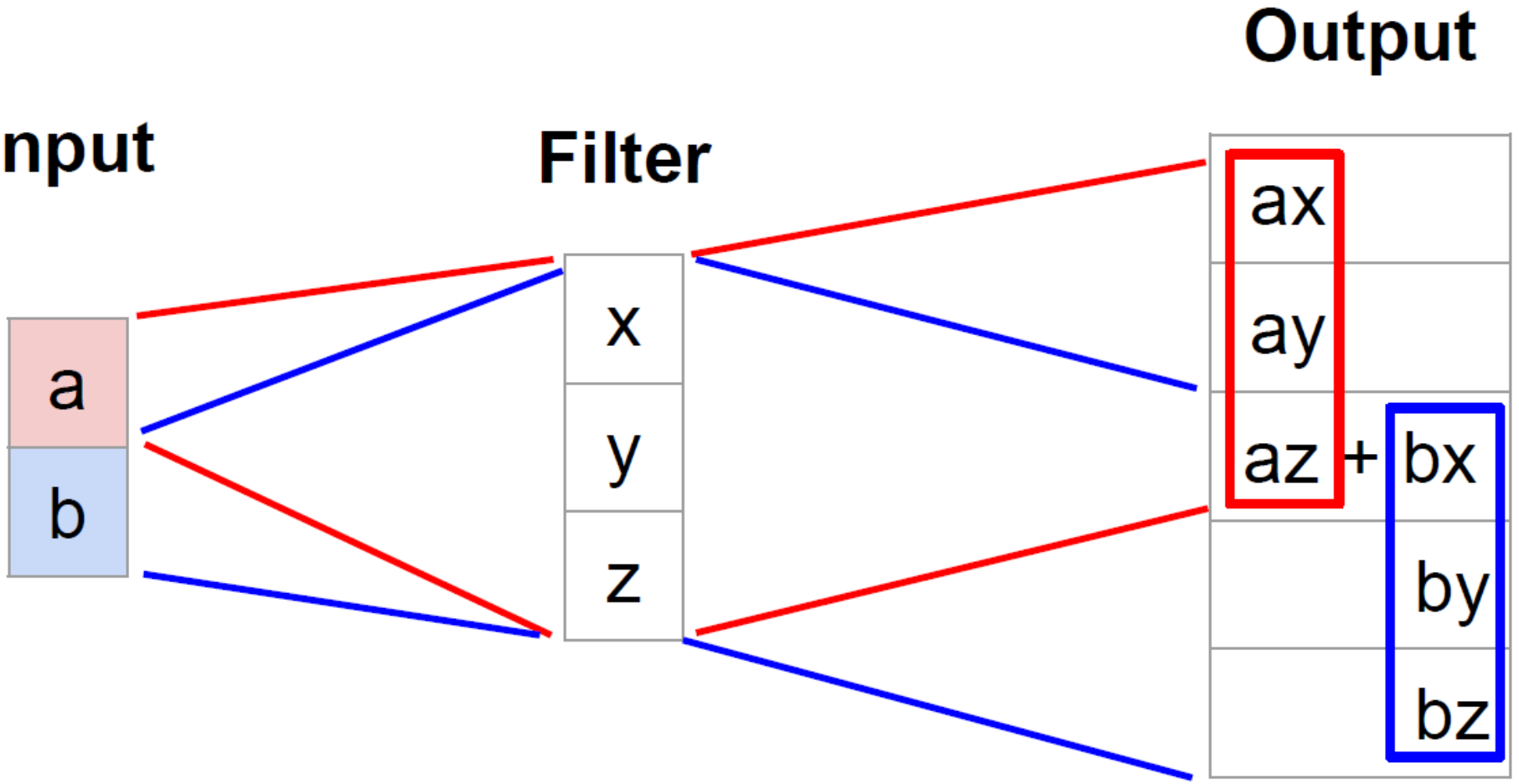


Figure 3. Illustration of deconvolution and unpooling operations.

Transpose Convolution: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

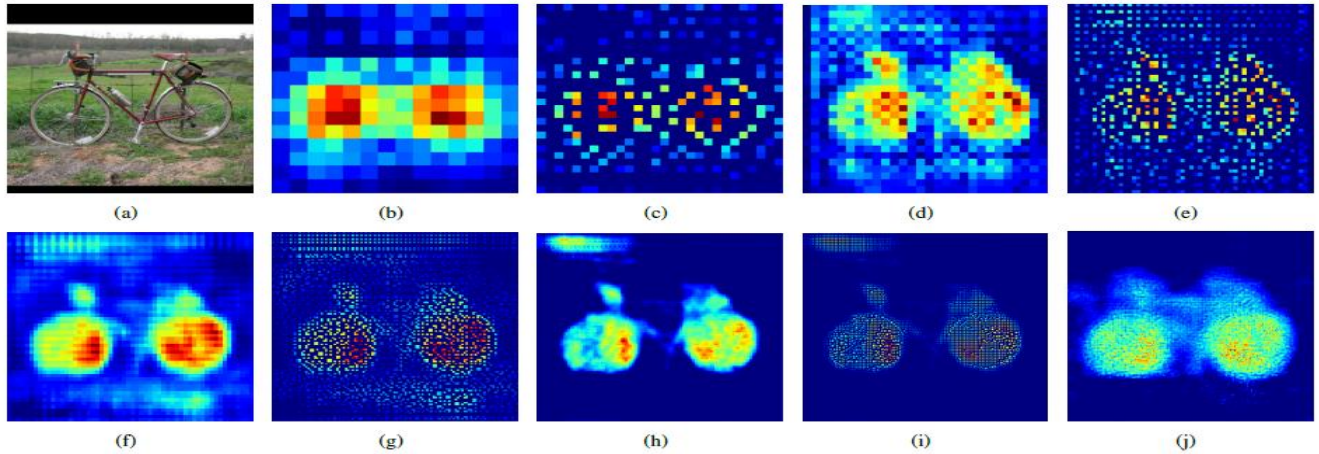
$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

3 System Architecture

3.2. Deconvolution Network for Segmentation

3.2.3 Analysis of Deconvolution Network



- Deconvolution layer에서 학습된 weight는 lower layer에서는 location과 전반적인 shape같은 coarse한 정보를 나타내고, upper layer에서는 complex detail을 나타낸다.

- Unpooling layer는 original location과 같은 example-specific structure를 잡아내고, Deconvolution layer는 class-specific shape를 잡아낸다.

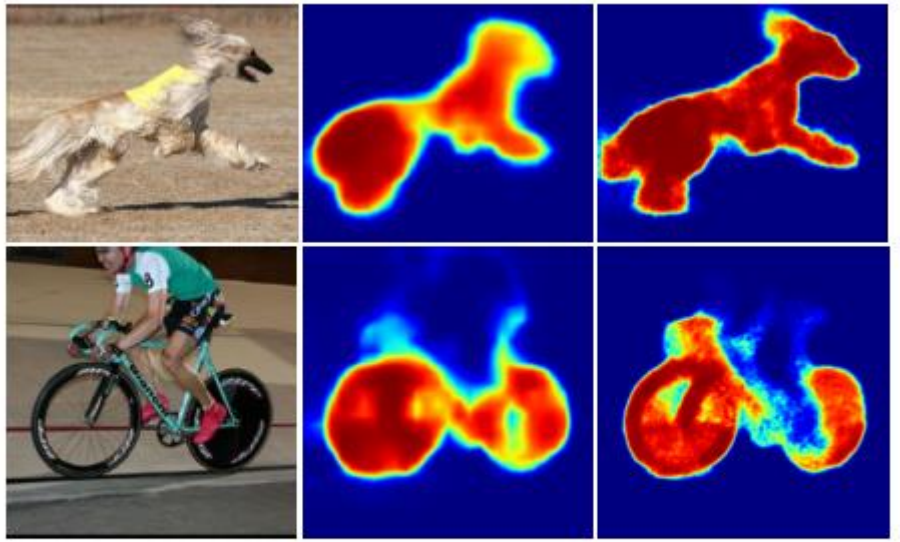
3 System Architecture

3.2. Deconvolution Network for Segmentation

3.2.3 Analysis of Deconvolution Network

- Deconvolution layer는 target-class와 관련된 activation은 증폭시키고, noisy는 억제한다.

- Unpooling과 Deconvolution layer의 조합으로 DeconvNet은 FCN보다 dense하고 정확한 segmentation 결과를 보여준다.



(a) Input image (b) FCN-8s (c) Ours

Figure 5. Comparison of class conditional probability maps from FCN and our network (top: dog, bottom: bicycle).

3 System Architecture

3.3. System Overview

- DeconvNet은 instance-wise segmentation으로 semantic segmentation 문제를 풀어낸다.
  즉, object를 포함한 sub-image인 candidate proposal을 input으로 받아들여 각각에 대한 pixel-wise class prediction output을 추출하고 이것을 종합해 semantic segmentation 결과를 추출한다.

- instance-wise segmentation은 fixed-size receptive field로 접근하는 image-level prediction과 다르게 다양한 scale을 다룰 수 있으며, object의 detail을 잘 인식할 수 있다.

4 Training

- DeconvNet은 FCN의 2배에 해당하는 깊이로 매우 deep하기 때문에 parameter수가 많은데,
  이에 비해 PASCAL VOC 2012 dataset은 상대적으로 작은 데이터이다.

- 따라서 이러한 문제점을 해결하기 위해 다음과 같은 방법들을 이용했다.

4 Training

4.1. Batch Normalization

- internal-covariate-shift 문제를 해결하기 위해 Batch Normalization을 이용한다.
- 모든 Convolutional layers와 Deconvolutional layers 뒤에 Batch Normalization layer를 적용한다.

4 Training

4.2. Two-stage Training

- DeconvNet은 FCN의 2배에 해당하는 깊이로 매우 deep하기 때문에 parameter수가 많은데, 이에 비해 dataset이 작으면 Deconvolution Network를 사용하는 것이 무의미해진다.

- 따라서, easy examples로 먼저 학습하고, 이 때 train된 network를 challenging example로 fine-tune하는 two-stage training을 통해 모델을 학습한다.

- First-stage example은 ground-truth bounding box가 중심에 오도록 image를 crop해서 생성한다.
  이렇게 생성된 example은 location과 size의 변동이 없기 때문에 parameter 추정이 상대적으로 쉬워진다.

- Second-stage example은 ground-truth bounding box와 충분히 겹치는 candidate proposal을 이용한다.
  candidate proposal은 location과 size가 다르기 때문에 challenging한 학습을 할 수 있으며, 모델을 robust 하게 만든다.

5 Inference

- DeconvNet으로 하나의 input image를 semantic segmentation하는 과정은 다음과 같다.

  Step 1) 여러 개의 candidate proposals 생성
  Step 2) DeconvNet으로 개별 instance(candidate proposal)에 대한 semantic segmentation을 수행
  Step 3) Step 2)의 결과를 통합해 whole image의 semantic segmentation 결과 도출
  Step 4) 선택적으로 성능 향상을 위해 FCN과의 ensemble 모델 이용

- Section5에서는 step 3)에 대한 자세한 과정을 설명한다.

5 Inference

5.1. Aggregating Instance-wise Segmentation Maps

- Aggregation을 통해 noisy를 제거하고 robust한 결과를 얻을 수 있다.

- 개별 instance(candidate proposal)에 대한 semantic segmentation 결과에 대해 pixel-wise maximum 또는 average를 취해 최종 scores map을 도출한다.

- Equation (1) or (2)를 통해 최종 pixel-wise class score map을 구하고,
  여기에 softmax를 취해 class conditional probability map을 구하고,
  마지막으로 fully-connected CRF를 적용해 pixel-wise labeling을 마무리 한다.

$$P(x, y, c) = \max_i G_i(x, y, c), \quad \forall i, \qquad (1)$$

$$P(x, y, c) = \sum_i G_i(x, y, c), \quad \forall i. \qquad (2)$$

5 Inference

5.2. Ensemble with FCN

- DeconvNet은 FCN과 상호보완적인 특징을 가진다.
  1) DeconvNet은 object의 fine-detail을 잘 잡아내는 반면에 FCN은 전반적인 shape를 잘 잡아낸다.
  2) DeconvNet은 instance-wise segmentation을 통해 다양한 scale의 object를 다룰 수 있지만,
     FCN은 single-scale만 다룰 수 있어서 이미지의 context를 잡아내기 좋다.

- 따라서, 이러한 상호보완적인 특징을 고려해 두 가지 모델을 ensemble한다.
  즉, 두 가지 모델의 output인 class conditional probability maps의 평균을 구하고, 그 결과에 CRF를 취해
  최종 semantic segmentation을 얻는다.

6 Experiments

6.1. Implementation Details

Network Configuration

: convolution network와 deconvolution network가 fc7 layer를 기준으로 symmetric한 구조를 가진다.
(input과 output의 depth만 다름)

Table 2. Detailed configuration of the proposed network. "conv" and "deconv" denote layers in convolution and deconvolution network, respectively, while numbers next to each layer name mean the order of the corresponding layer in the network. ReLU layers are omitted from the table for brevity.

name	kernel size	stride	pad	output size
input	-	-	-	$224 \times 224 \times 3$
conv1-1	3×3	1	1	$224 \times 224 \times 64$
conv1-2	3×3	1	1	$224 \times 224 \times 64$
pool1	2×2	2	0	$112 \times 112 \times 64$
conv2-1	3×3	1	1	$112 \times 112 \times 128$
conv2-2	3×3	1	1	$112 \times 112 \times 128$
pool2	2×2	2	0	$56 \times 56 \times 128$
conv3-1	3×3	1	1	$56 \times 56 \times 256$
conv3-2	3×3	1	1	$56 \times 56 \times 256$
conv3-3	3×3	1	1	$56 \times 56 \times 256$
pool3	2×2	2	0	$28 \times 28 \times 256$
conv4-1	3×3	1	1	$28 \times 28 \times 512$
conv4-2	3×3	1	1	$28 \times 28 \times 512$
conv4-3	3×3	1	1	$28 \times 28 \times 512$
pool4	2×2	2	0	$14 \times 14 \times 512$
conv5-1	3×3	1	1	$14 \times 14 \times 512$
conv5-2	3×3	1	1	$14 \times 14 \times 512$
conv5-3	3×3	1	1	$14 \times 14 \times 512$
pool5	2×2	2	0	$7 \times 7 \times 512$
fc6	7×7	1	0	$1 \times 1 \times 4096$
fc7	1×1	1	0	$1 \times 1 \times 4096$
deconv-fc6	7×7	1	0	$7 \times 7 \times 512$
unpool5	2×2	2	0	$14 \times 14 \times 512$
deconv5-1	3×3	1	1	$14 \times 14 \times 512$
deconv5-2	3×3	1	1	$14 \times 14 \times 512$
deconv5-3	3×3	1	1	$14 \times 14 \times 512$
unpool4	2×2	2	0	$28 \times 28 \times 512$
deconv4-1	3×3	1	1	$28 \times 28 \times 512$
deconv4-2	3×3	1	1	$28 \times 28 \times 512$
deconv4-3	3×3	1	1	$28 \times 28 \times 256$
unpool3	2×2	2	0	$56 \times 56 \times 256$
deconv3-1	3×3	1	1	$56 \times 56 \times 256$
deconv3-2	3×3	1	1	$56 \times 56 \times 256$
deconv3-3	3×3	1	1	$56 \times 56 \times 128$
unpool2	2×2	2	0	$112 \times 112 \times 128$
deconv2-1	3×3	1	1	$112 \times 112 \times 128$
deconv2-2	3×3	1	1	$112 \times 112 \times 64$
unpool1	2×2	2	0	$224 \times 224 \times 64$
deconv1-1	3×3	1	1	$224 \times 224 \times 64$
deconv1-2	3×3	1	1	$224 \times 224 \times 64$
output	1×1	1	1	$224 \times 224 \times 21$

6 Experiments

6.1. Implementation Details

Dataset

: PASCAL VOC 2012 segmentation dataset을 이용했으며,
다른 모델들과 다르게 그 외 dataset을 사용하지 않고 data augmentation을 이용한다.

Training Data Construction

: 1) first-stage data는 object에 tight한 bounding-box를 그린 후, 그것을 1.2배 확대해 배경을 포함한 example을 만든다.
2) second-stage data는 1)에서 생성한 data와 Edge-Box로 생성한 2000개의 region proposals 중 objectness score가 높은 상위 50개의 proposals를 250x250 size로 만들고 random하게 224x224 size로 crop해 data augmentation을 한다.

6 Experiments

6.1. Implementation Details

Optimization

- momentum optimizer를 이용했으며, initial learning rate=0.01, momentum=0.9, weight decay=0.0005로 설정하고, validation accuracy가 증가하지 않으면 learning rate를 감소시킨다.

- Convolution Network의 weight 초기값은 pre-trained VGG16으로 설정하고, Deconvolution Network의 weight는 zero-mean Gaussian으로 설정한다.

- Dropout 없이 Batch Normalization만 이용한다.

- 64의 mini-batch size로 first-stage, second-stage를 각각 20K, 40K iteration만큼 train한다. (6일 소요)

6 Experiments

6.2. Evaluation on Pascal VOC

- ground-truth와 predicted segmentation의 IOU로 점수를 측정하는 comp6 evaluation을 적용한 결과이다.
- FCN과의 ensemble 모델이 72.5%로 가장 높은 accuracy를 보여준다.

Table 1. Evaluation results on PASCAL VOC 2012 test set. (Asterisk (∗) denotes the algorithms trained with additional data.)

Method	bkg	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbk	person	plant	sheep	sofa	train	tv	mean
Hypercolumn [10]	88.9	68.4	27.2	68.2	47.6	61.7	76.9	72.1	71.1	24.3	59.3	44.8	62.7	59.4	73.5	70.6	52.0	63.0	38.1	60.0	54.1	59.2
MSRA-CFM [3]	87.7	75.7	26.7	69.5	48.8	65.6	81.0	69.2	73.3	30.0	68.7	51.5	69.1	68.1	71.7	67.5	50.4	66.5	44.4	58.9	53.5	61.8
FCN8s [17]	91.2	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
TTI-Zoomout-16 [18]	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
DeepLab-CRF [1]	93.1	84.4	**54.5**	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	**59.7**	82.2	50.4	73.1	63.7	71.6
DeconvNet	92.7	85.9	42.6	78.9	62.5	66.6	87.4	77.8	79.5	26.3	73.4	60.2	70.8	76.5	79.6	77.7	58.2	77.4	52.9	75.2	59.8	69.6
DeconvNet+CRF	92.9	87.8	41.9	80.6	63.9	67.3	**88.1**	78.4	81.3	25.9	73.7	61.2	72.0	77.0	79.9	78.7	59.5	78.3	**55.0**	75.2	61.5	70.5
EDeconvNet	92.9	88.4	39.7	79.0	63.0	67.7	87.1	**81.5**	84.4	27.8	76.1	61.2	78.0	79.3	83.1	79.3	58.0	82.5	52.3	80.1	64.0	71.7
EDeconvNet+CRF	93.1	**89.9**	39.3	79.7	63.9	**68.2**	87.4	81.2	**86.1**	28.5	**77.0**	62.0	79.0	**80.3**	**83.6**	80.2	58.8	**83.4**	54.3	**80.7**	65.0	**72.5**
* WSSL [19]	93.2	85.3	36.2	**84.8**	61.2	67.5	84.7	81.4	81.0	**30.8**	73.8	53.8	77.5	76.5	82.3	**81.6**	56.3	78.9	52.3	76.6	63.3	70.4
* BoxSup [2]	**93.6**	86.4	35.5	79.7	**65.2**	65.2	84.3	78.5	83.7	30.5	76.2	**62.6**	**79.3**	76.1	82.1	81.3	57.0	78.2	**55.0**	72.5	**68.1**	71.0

6 Experiments

6.2. Evaluation on Pascal VOC

- Proposals의 개수가 증가할수록 segmentation이 정확해지는 것으로 보아 instance-wise prediction이 정확한 segmentation에 효과가 있음을 알 수 있다.

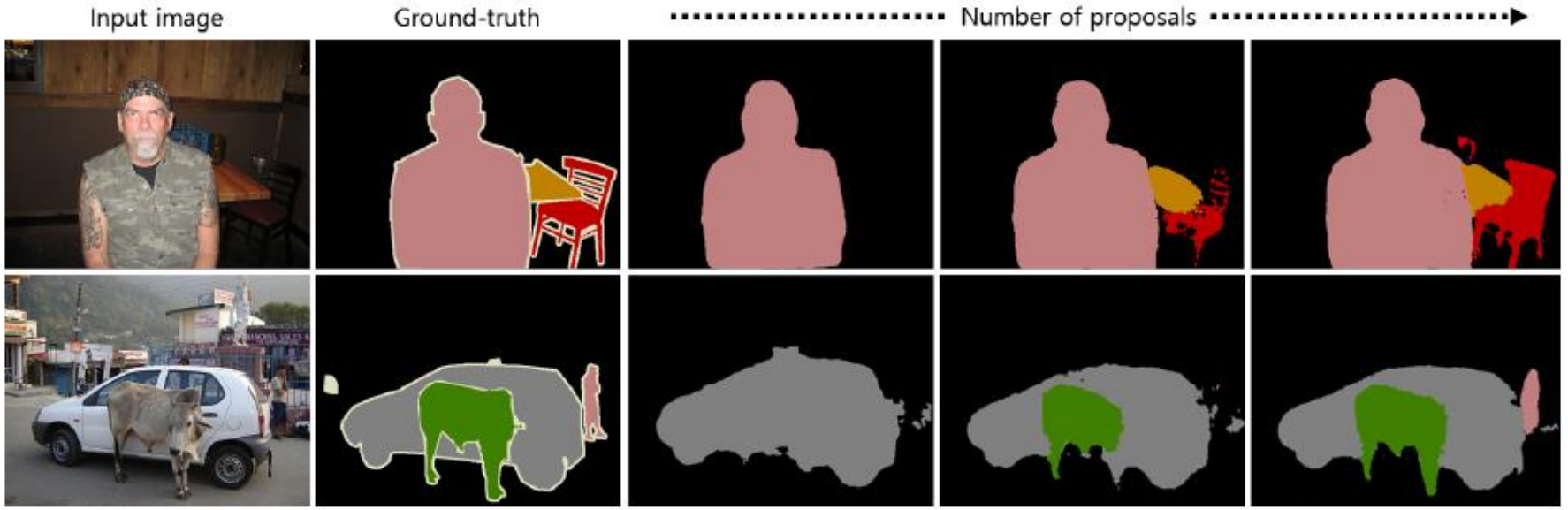


Figure 6. Benefit of instance-wise prediction. We aggregate the proposals in a decreasing order of their sizes. The algorithm identifies finer object structures through iterations by handling multi-scale objects effectively.