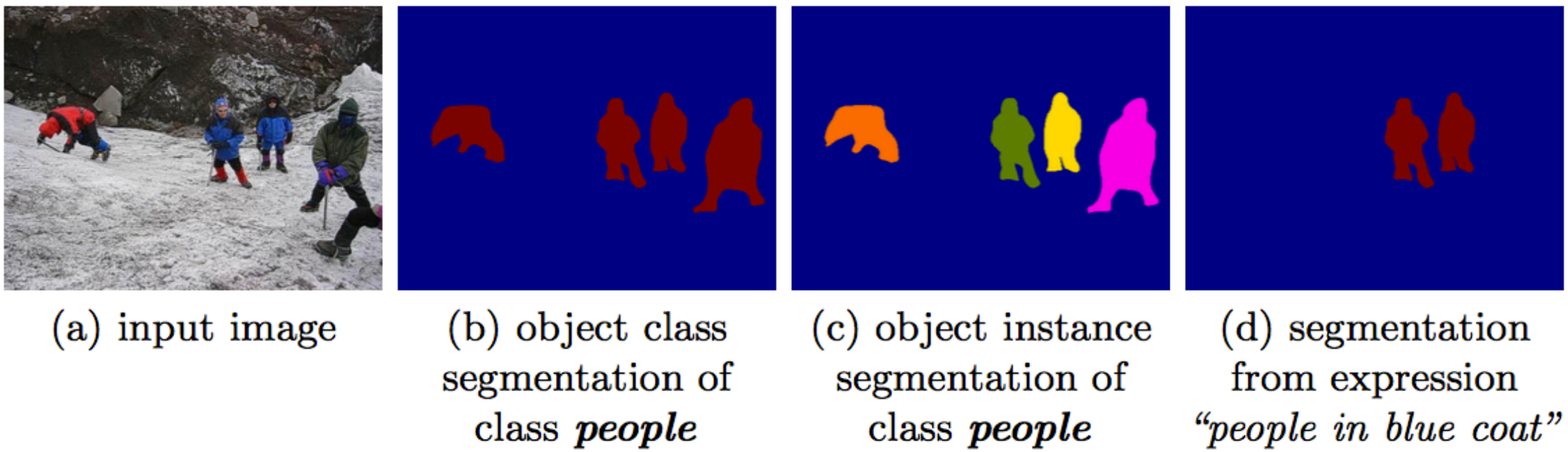
Segmentation from Natural Language Expressions

Ronghang Hu, Marcus Rohrbach, Trevor Darrell

Presenter: Tianyi Jin

Comparisons between different semantic image segmentation problems



(a) input image

(b) object class segmentation of class ***people***

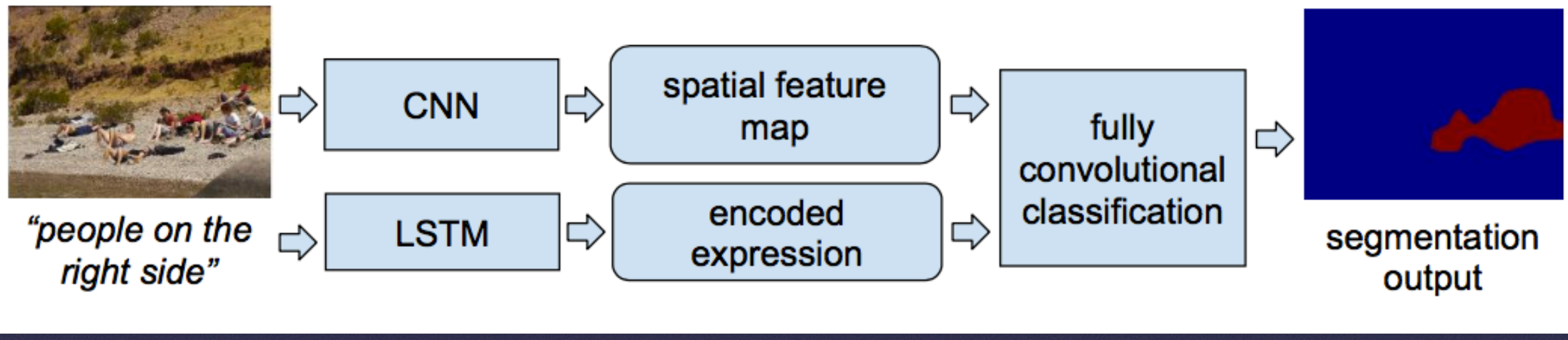
(c) object instance segmentation of class ***people***

(d) segmentation from expression *"people in blue coat"*

(e) Grabcut: generate a mask over the foreground (or the most salient) object

(f) Natural Language Object Retrieval: bounding box only, non pixelwise

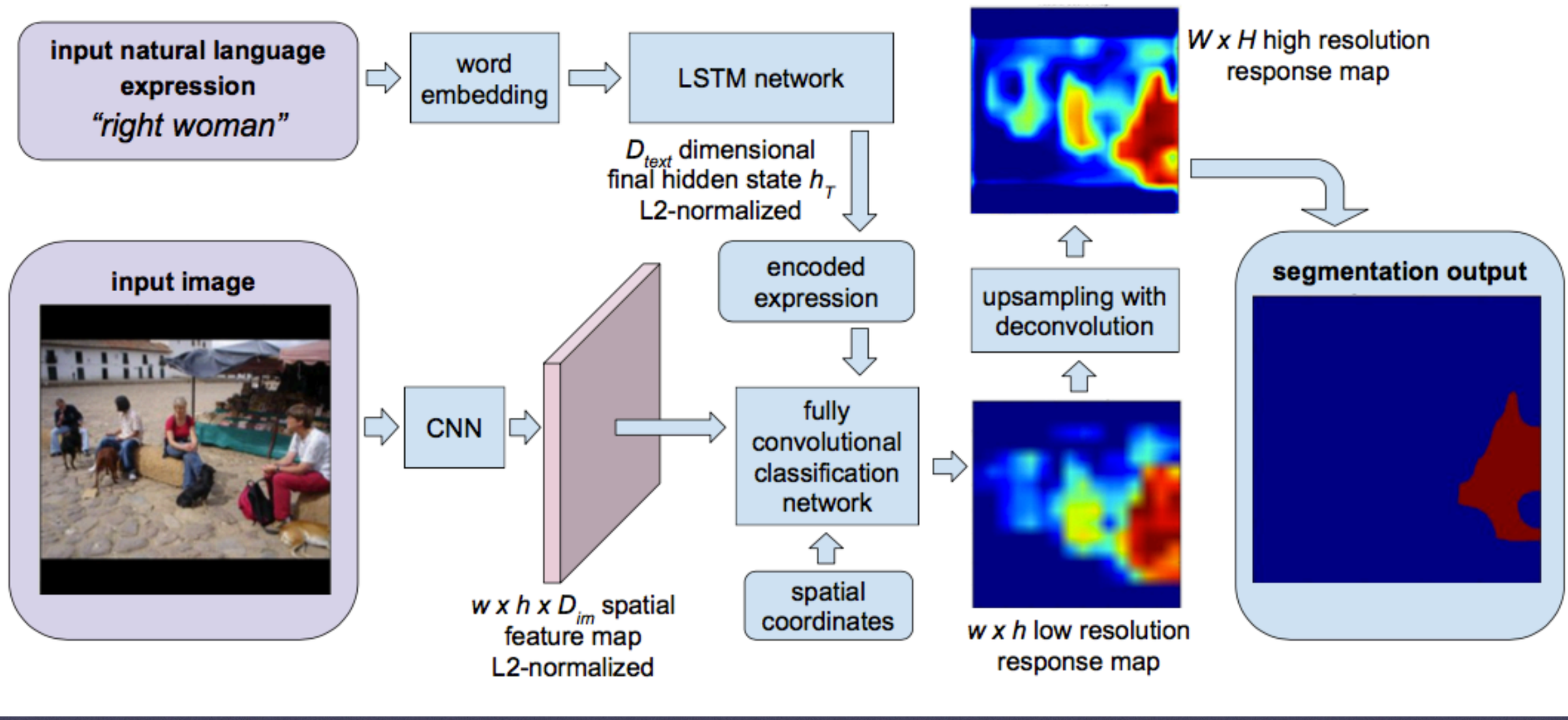
Overview



Goal: Pixel-level segmentation of image, based on natural language expression

Related Work

- Localizing objects with natural language
  - bounding box only
- Fully convolutional network for segmentation
  - used for feature extraction and segmentation output
- Attention and visual question answering
  - only learn to generate coarse spatial outputs, with other purposes



Our Model

A Detailed Look At 👀

Spatial feature map extraction

- Fully convolutional network
  - Input image size: *W x H*, spatial feature map size: *w × h*, with each position on the feature map containing *Dim* channels (*Dim* dimensional local descriptors)
- Apply L2-normalization to the *Dim* dimensional local descriptor
  - Extract a *w × h × Dim* spatial feature map as the representation for each image
- Add extra channels of *x, y* coordinate of each spatial location
  - Get a *w × h × (Dim+2)* representation containing descriptors and spatial coordinates
- In this implementation: VGG-16 with treating fc6, fc7 and fc8 as convolutional layers, which outputs *Dim = 1000* dimensional local descriptors.
- Resulting feature map size: *w = W/s and h = H/s*, where *s = 32* is the pixel stride on fc8 layer output. (Here *W = H = 512*)

Encoding expressions with LSTM network

- Embed each word into a vector through a word embedding matrix
- Use a recurrent Long-Short Term Memory (LSTM) network with *Dtext* dimensional hidden state to scan through the embedded word sequence
- L2-normalize
- In this implementation: LSTM network with *Dtext = 1000* dimensional hidden state

Spatial classification and upsampling

- Fully convolutional classifier over the local image descriptor and the encoded expression
  - Tile and concatenate hidden state to the local descriptor at each spatial location in the spatial grid -> a *w×h×D'* (where *D' = Dim +Dtext +2*) spatial map
  - Train a two-layer classification network (two *1 x 1* convolutional layers), with a D_{cls} dimensional hidden layer, which takes at input the *D'* dimensional representation -> a score indicating whether a spatial location belong to the target image region or not
  - In this implementation: $D_{cls} = 500$
- Upsampling through deconvolution
  - a $2s \times 2s$ deconvolution filter with stride s (here $s = 32$)
  - Produces a $W \times H$ high resolution response map that has the same size as the input image

Loss Function

At training time, each training instance in our training set is a tuple (I, S, M), where I is an image, S is a natural language expression describing a region within that image, and M is a binary segmentation mask of that region. The loss function during training is defined as the average over pixelwise loss

$$Loss = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} L(v_{ij}, M_{ij}) \tag{1}$$

where W and H are image width and height, v_{ij} is the response value (score) on the high resolution response map and M_{ij} is the binary ground-truth label at pixel (i, j). L is the per-pixel weighed logistic regression loss as follows

$$L(v_{ij}, M_{ij}) = \begin{cases} \alpha_f \log(1 + \exp(-v_{ij})) & \text{if } M_{ij} = 1 \\ \alpha_b \log(1 + \exp(v_{ij})) & \text{if } M_{ij} = 0 \end{cases} \tag{2}$$

where α_f and α_b are loss weights for foreground and background pixels. In practice, we find that training converges faster using higher loss weights for foreground pixels, and we use $\alpha_f = 3$ and $\alpha_b = 1$ in $L(v_{ij}, M_{ij})$.

Experiments

- Dataset: ReferIt [1]
- 20,000 images. 130,525 expressions annotated on 96,654 segmented image regions.
  - Here: 10,000 images for training and validation,10,000 images for testing
  - contains both “object” regions (car, person, bottle) and “stuff” regions (sky, river and mountain)
- Baseline methods
  - Combination of per-word segmentation
  - Foreground segmentation from bounding boxes
  - Classification over segmentation proposals
  - Whole image

[1] Kazemzadeh, S., Ordonez, V., Matten, M., Berg, T.L.: Referitgame: Referring to objects in photographs of natural scenes (EMNLP 2014)

Evaluation

- Two-stage training strategy:
  - Low resolution version: $w \times h = 16 \times 16$ coarse response map
  - High resolution version: upsampled from low resolution model, predict $W \times H$ high resolution segmentation
- Overall IoU: total intersection area divided by the total union area
- Precision: the percentage of test samples where the IoU between prediction and ground-truth passes the threshold

Results

Method	*prec*@0.5	*prec*@0.6	*prec*@0.7	*prec*@0.8	*prec*@0.9	overall IoU
whole image	5.07%	2.85%	1.58%	0.81%	0.41%	15.12%
per-word average	10.97%	5.94%	2.35%	0.45%	0.00%	27.23%
per-word intersection	9.58%	5.35%	2.20%	0.43%	0.00%	26.69%
per-word union	10.46%	5.65%	2.28%	0.44%	0.00%	19.37%
SCRC [11] bbox	9.73%	4.43%	1.51%	0.27%	0.03%	21.72%
SCRC [11] grabcut	11.91%	7.71%	4.33%	1.78%	0.36%	17.84%
GroundeR [12] bbox	11.08%	6.20%	2.74%	0.78%	0.20%	20.50%
GroundeR [12] grabcut	14.09%	9.62%	5.78%	2.65%	0.62%	20.09%
MCG classification	12.72%	9.88%	7.38%	4.73%	1.88%	18.08%
Ours (low resolution)	29.54%	21.61%	13.69%	5.94%	0.75%	45.57%
Ours (high resolution)	**34.02%**	**26.71%**	**19.32%**	**11.63%**	**3.92%**	**48.03%**

Table 1. The performance of our model and baseline methods on the ReferIt dataset under precision metric and overall IoU metric. See Section 4 for details.

Method	per-word	SCRC [11] grabcut	GroundeR [12] grabcut	MCG classification	Ours (high resolution)
time (sec)	0.169	4.319	3.753	9.375	0.325

Table 2. Average time consumption to segmentation an input (a given image and a natural language expression) using different methods.

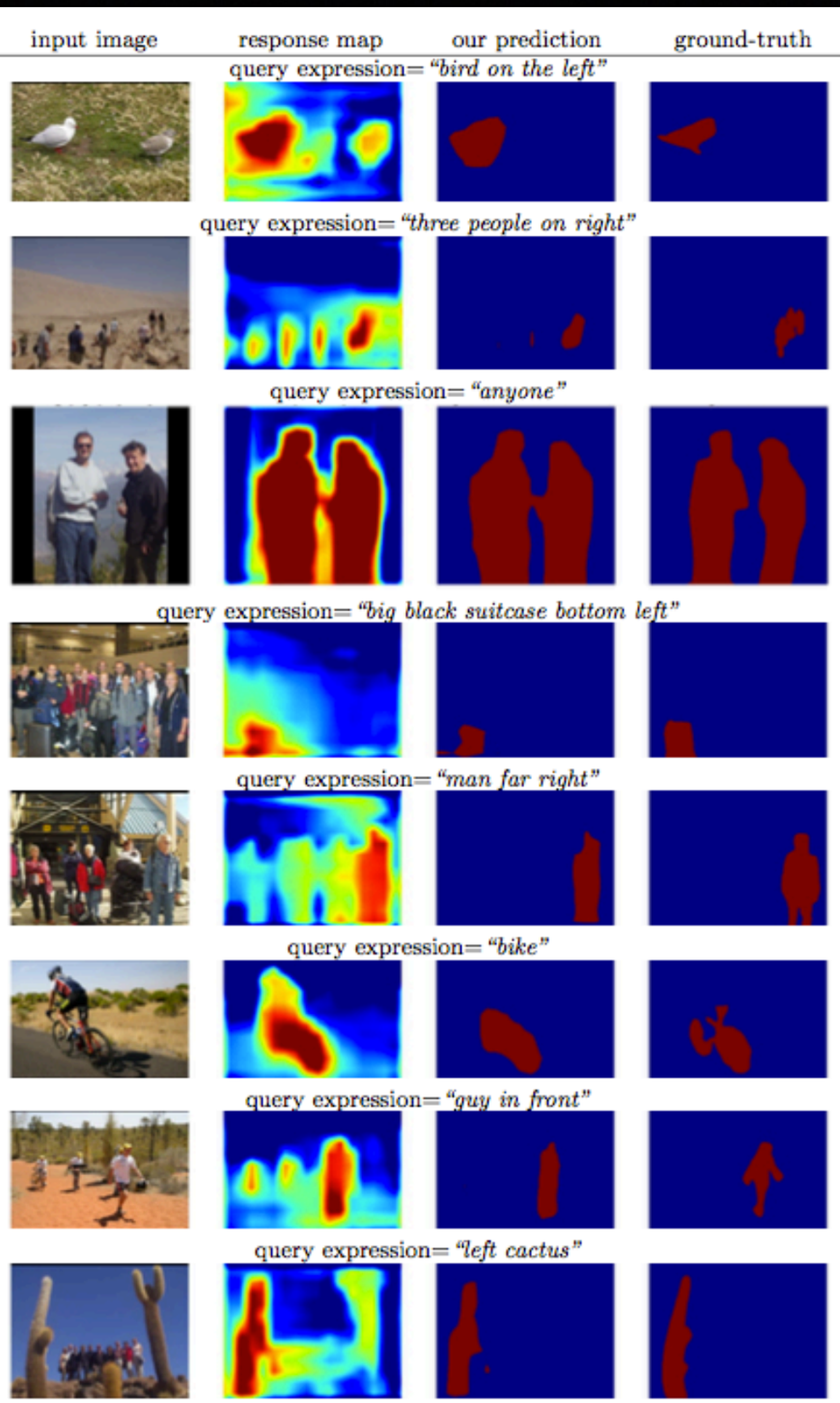


Fig. 5. Segmentation examples on object regions in the ReferIt dataset.

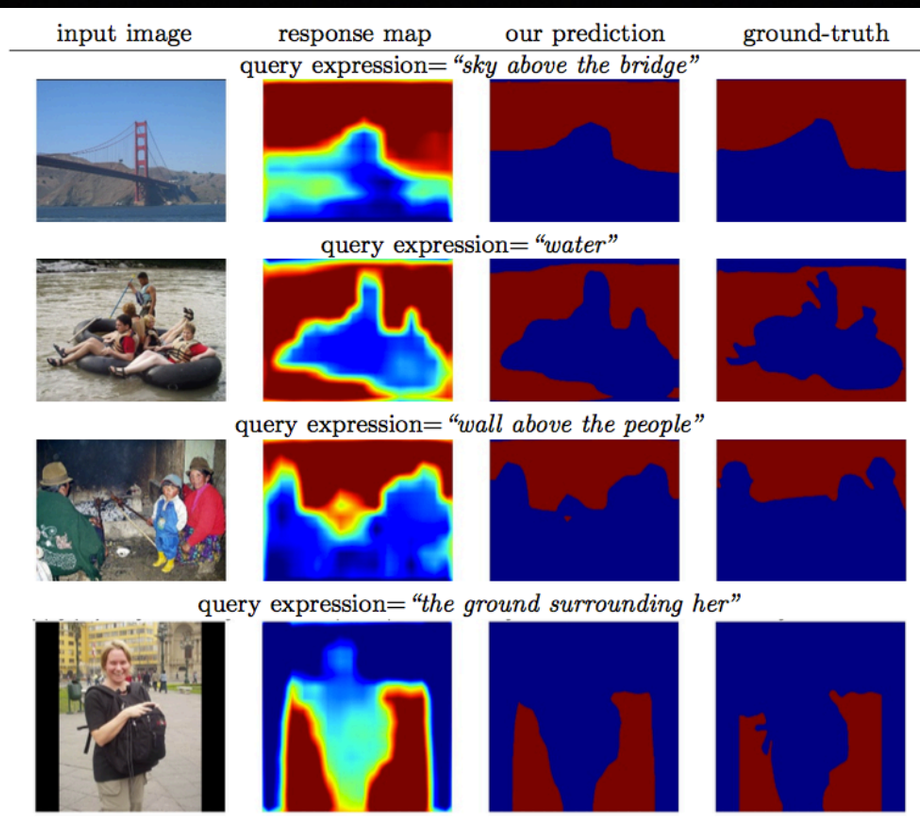


Fig. 6. Segmentation examples on stuff regions in the ReferIt dataset.

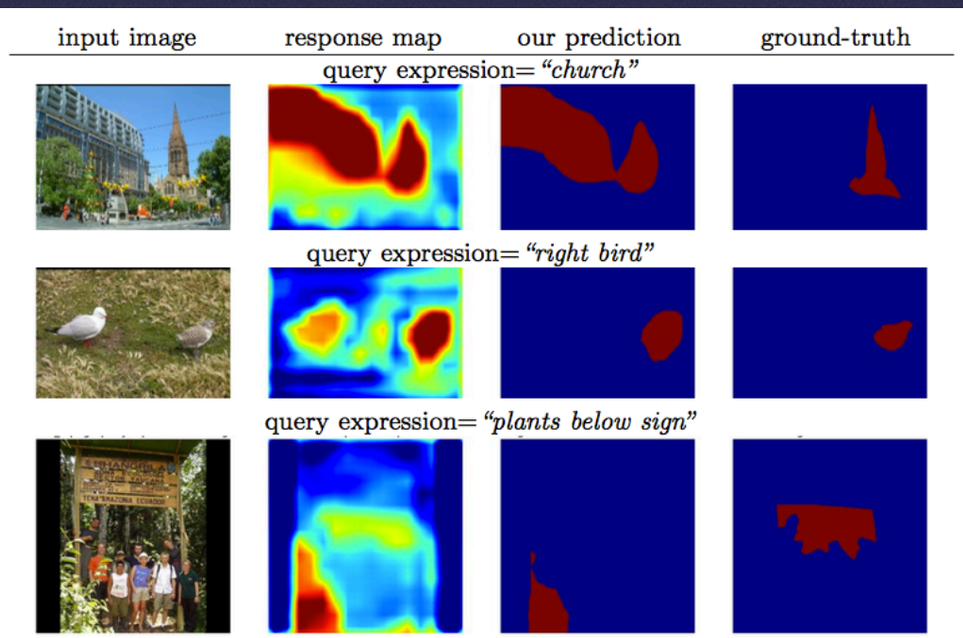
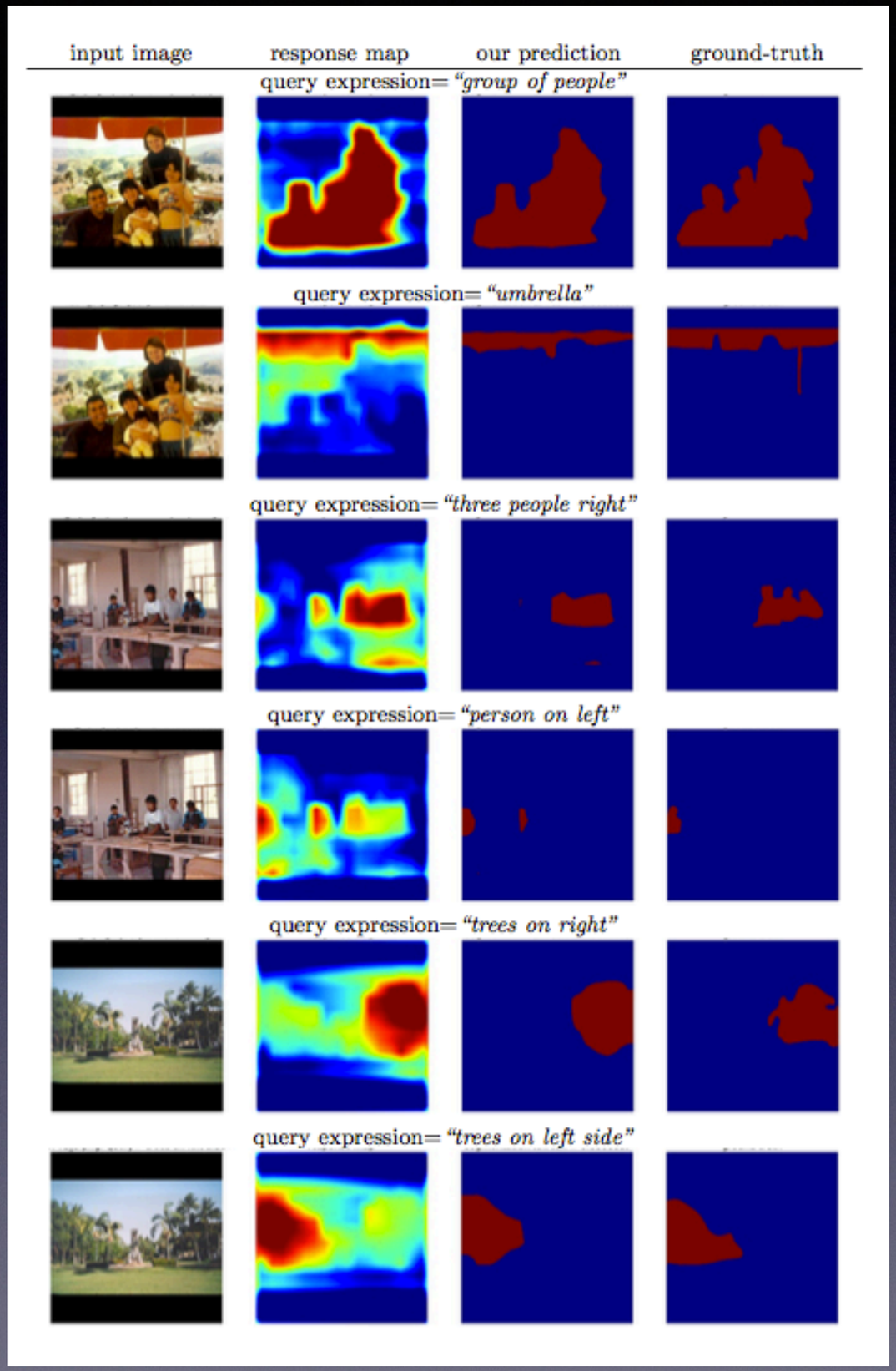
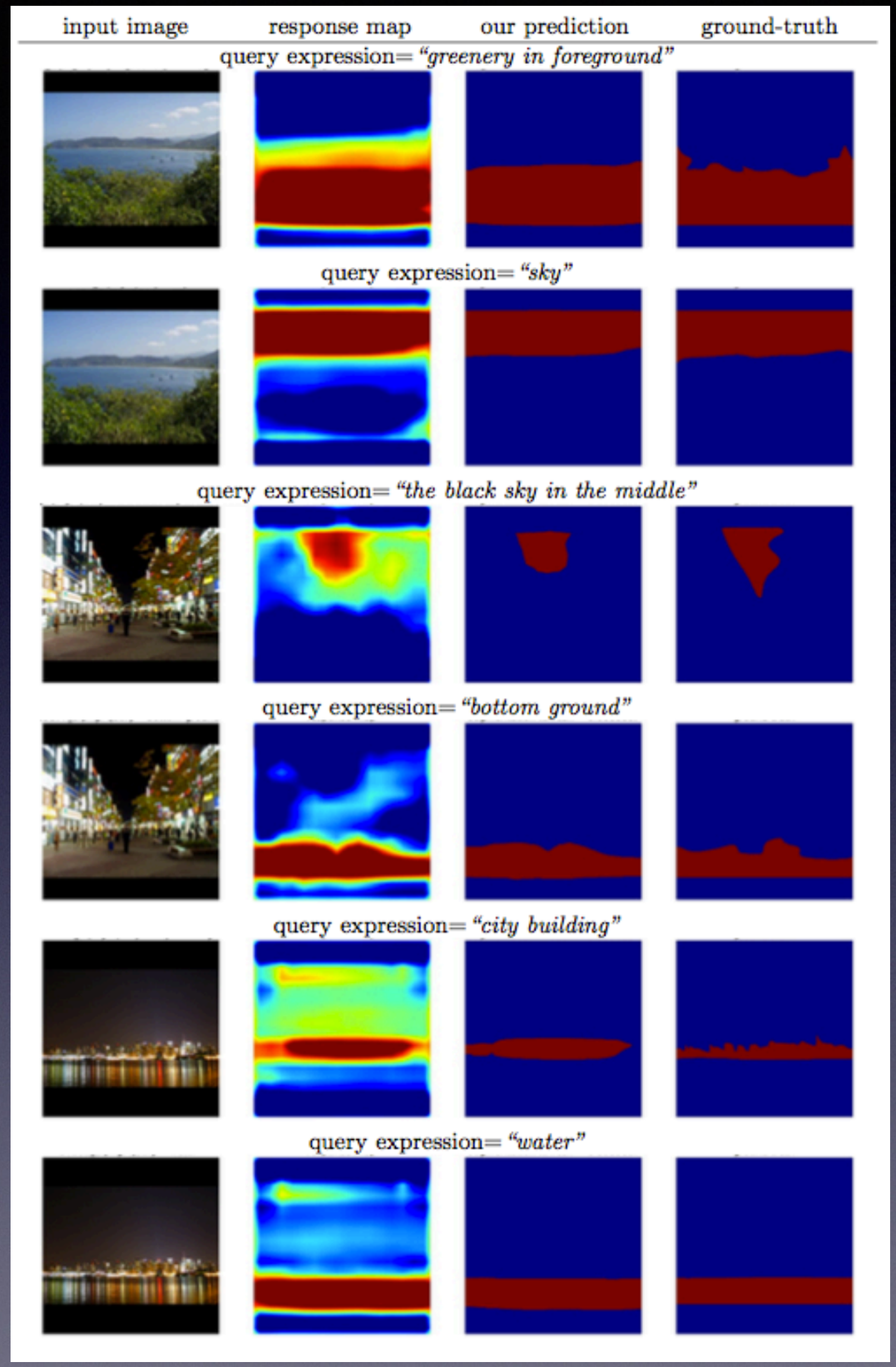


Fig. 7. Some failure cases where IoU $< 50\%$ between prediction and ground-truth.


input image
response map
our prediction
ground-truth
query expression= "group of people"
query expression= "umbrella"
query expression= "three people right"
query expression= "person on left"
query expression= "trees on right"
query expression= "trees on left side"


input image
response map
our prediction
ground-truth
query expression= "greenery in foreground"
query expression= "sky"
query expression= "the black sky in the middle"
query expression= "bottom ground"
query expression= "city building"
query expression= "water"

Questions?

Thank you!