

Object segmentation defined by Query expressions

Ravi Choudhary, Aftaab Mohammed
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst

ABSTRACT:

In this project we are trying to solve a object segmentation with query expression problem using visual and linguistic features derived from the given image and its text expression. The main approach involves deep learning techniques like Convolution neural networks (CNN) to extract spatial feature map from the image and LSTM (Long short term memory networks)- special kind of RNN (Recurrent neural network) used to encode given expression . Our proposed architecture is then trained on a widely available dataset proving the results outperform baseline algorithms giving IOU accuracy near 47%.

INTRODUCTION:

The idea was basically inspired by the core problem in computer vision, image segmentation (you will recognize all objects in an image and your output should show this object classifying pixels of the image) and extending this problem by using a query expression defining which object to segment out in the given image. For example, in the below given Figure 1, the query expression defines “The water at the bottom of the picture” so the idea is to use this phrase expression and pixel-wise segment out what it conveys from the picture. Task of segmenting an image from natural language expressions has a wide range of applications, such as building language-based human-robot interface to give instructions like “pick up the jar on the table next to the apples” to a robot. Here, it is important to be able to use multi-word query expressions to distinguish between different object instances but also important to get a precise segmentation problem of segmenting an image based on an expression.

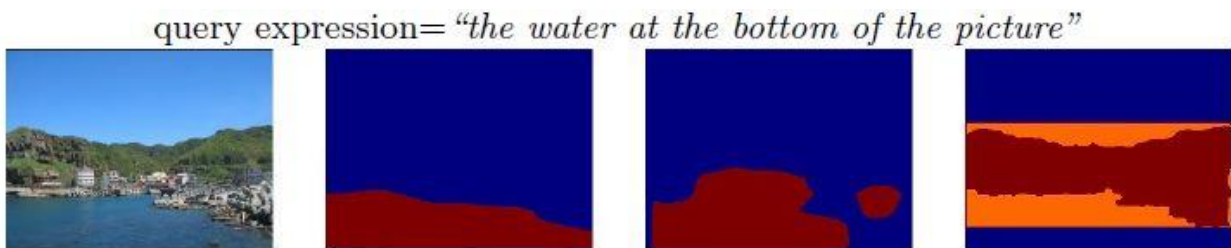


Fig 1: Different pictures from the right representing 3 different models segmented output.

The goal of the project is to output a segmentation mask for the visual entities described by the expression. To accomplish this goal, we propose a model with three main components: a natural language expression encoder based on a recurrent LSTM network i.e. the most

important part of our project, a fully convolution network to extract local image descriptors and generate a spatial feature map, and a fully convolution classification and up-sampling network that takes as input the encoded expression and the spatial feature map and outputs a pixel-wise segmentation mask.

Semantic image segmentation is a core problem in computer vision and significant progress has been made using large visual datasets and rich representations based on convolution neural networks. Although these existing segmentation methods can predict precise pixel-wise masks for query categories like “train” or “cat”, they are not capable of predicting segmentation for more complicated queries such as the query expression “the two people on the right side of the car wearing black shirt” and that is what the main research problem we are trying to solve.

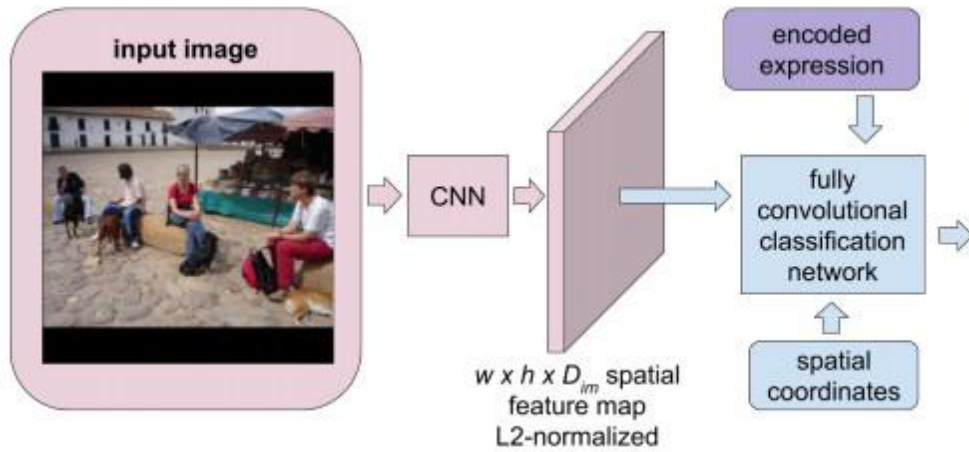


Fig 2: Representing Architecture of our model

Related work:

The methods reported in [11] build upon image captioning frameworks such as LRCN [3] or mRNN [4], and localize objects by selecting the bounding box where the expression has the highest probability. Our model differs from [2] in that we do not have to learn to generate expressions from image regions. In [12], the authors propose a model to localize a textual phrase by attending to a region on which the phrase can be best reconstructed. In [1], Canonical Correlation Analysis (CCA) is used to learn a joint embedding space of visual features and words, and given a natural language query; the corresponding target object is localized by finding the closest region to the text sequence in the joint embedding space. Fully convolution networks are convolution neural networks consisting of only convolution (and pooling) layers, which are the state-of-the-art method for semantic segmentation over a pre-defined set of semantic categories [6,7,8,9]. A nice property of fully convolution networks is that spatial information is preserved in the output, which makes these networks suitable for segmentation tasks that require spatial grid output. In our model, both feature extraction and segmentation output is performed through fully convolution networks. In paper [13] the image feature

descriptors used was a modified version of CNN -16 layer architecture we were using, where they used fully convolution networks instead of fully connected layers at the end. In [12] localizing a word or phrase in a given image, an approach which can learn to localize phrases relying only on phrases associated with images without bounding box annotations was implemented.

Data Sets:

There are only few publicly available datasets with query expressions defined over segmented image regions. In our experiments, we train and test our method on the ReferIt dataset [5] which has 20,000 images. There are 130,525 expressions annotated on 96,654 segmented image regions (some regions are annotated with multiple expressions). The expressions in the ReferIt dataset [5] are discriminative for the regions and goal was to make the target region easily distinguishable through the expression from the rest of the image. It is the biggest publicly available dataset that contains natural language expressions annotated on segmented image regions. We have split the data as 10,000 images for training and validation, and 10,000 images for testing. The annotated regions in the ReferIt dataset contain both object regions such as car, person and bottle and stuff regions such as sky, river and mountain.

Model:

We Implemented a model with three main components: a natural language expression encoder based on a recurrent LSTM network i.e. the most important part of our project, a fully convolution network to extract local image descriptors and generate a spatial feature map, and a fully convolution classification and up-sampling network that takes as input the encoded expression and the spatial feature map and outputs a pixel-wise segmentation mask.

1. VGG-16 for spatial feature map:

So let's first begin with the initial component i.e. fully convolution network to extract Image descriptors. The reason we thought to use Convolutional networks (ConvNets) mainly was because of its great success in large-scale image and video recognition. So we started to check around some of the start of the art architecture which could be helpful and relative to our proposed model. We came across this paper [14] by guys at oxford university, where they investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. This paper has some solid work, evidence and also the presented model was winning architecture at ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). Before going further with this architecture we checked around some more papers like [3] which were one of the important contributions in large scale image recognition, alas we couldn't find much more in depth details of this architecture so we went ahead with [14] and tried to implement it. [1] Presented about 6 different weight layer frameworks, we designed the most appreciated 16 weight layer design and in next steps, the idea would be to train our images and give out some interesting feature descriptors for the given training images.

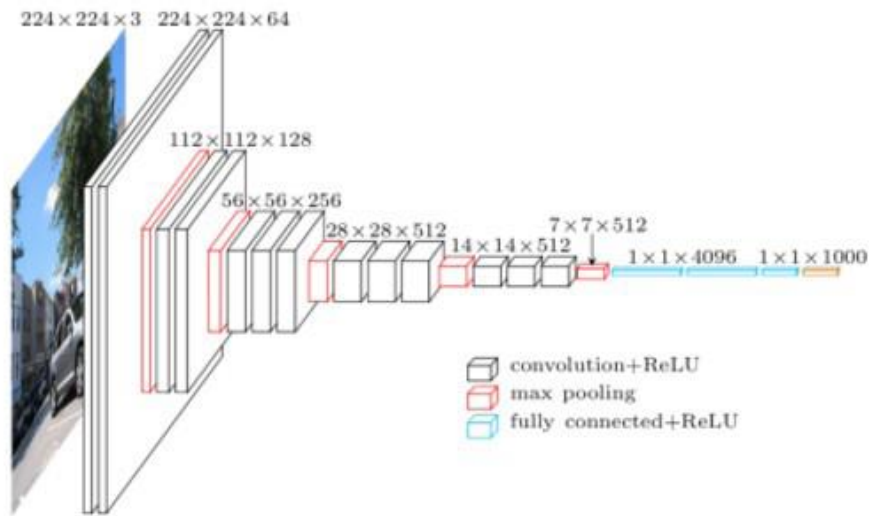


Fig 3: Architecture of VGG-16 Network as explained above

1.1 Architecture:

During training, the input to our ConvNets is a fixed-size RGB image, from the figure[1] below, the only preprocessing we thought do is subtracting the mean RGB value, computed on the training set, from each pixel as deep learning networks traditionally share many parameters - if you didn't scale your inputs in a way that resulted in similarly-ranged feature values (ie: over the whole dataset by subtracting mean) sharing wouldn't happen very easily because to one part of the image weight is a lot and to another it's too small. Below figure[1] shows exactly what is explained below. The image is passed through a stack of convolution (conv.) layers, where we use filters with a very small receptive field: 3×3 . The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2. A stack of convolution layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way classification and thus contains 1000 channels (one for each class). This architecture was implemented using Tensorflow and python. As mentioned above this is what the progress was until now for this spatial feature descriptors section, next steps will include trying to train our images from ReferIt dataset, also to allow the model to reason about spatial relationships such as right woman two extra channels may be added to the feature maps: the x and y coordinate of each spatial location. The paper [14] from which the architecture was adopted from, had about 6 different formats with increasing depth. The table 1 below shows in details what these formats were, due to the popularity of the VGG-16 (column D in table below)

format and assuming this may be the right architecture for our model, we have adopted and implemented it.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 1: Different architectures designed by the Visual geometry group

Given an input image of size $W \times H$, we use a convolution network explained above on the image to obtain a $w \times h$ spatial feature map, with each position on the feature map containing Dim channels (Dim dimensional local descriptors). For each spatial location on the feature map, we apply L2-normalization to the Dim dimensional local descriptor at that position in order to obtain a more robust feature representation. In this way, we can extract a $w \times h \times \text{Dim}$ spatial feature map as the representation for each image. Also, to allow the model to reason about spatial relationships such as “right woman”, two extra channels are added to the feature maps: the x and y coordinate of each spatial location. We use relative coordinates, where the upper left corner and the lower right corner of the feature map are represented as $(-1, -1)$ and $(+1, +1)$, respectively. In this way, we obtain a $w \times h \times (\text{Dim}+2)$ representation containing local image descriptors and spatial coordinates. The resulting feature map size is $w = W/s$ and $h = H/s$, where $s = 32$ is the pixel stride on fc8 layer output. The above little tricks were inspired by [13]

paper, where they use their CNN architecture with L2 normalization and adding spatial coordinates tricks gave them better accurate image descriptors calculation.

2. A Natural Language Expression encoder:

2.1 Baseline Algorithm (Bag of words):

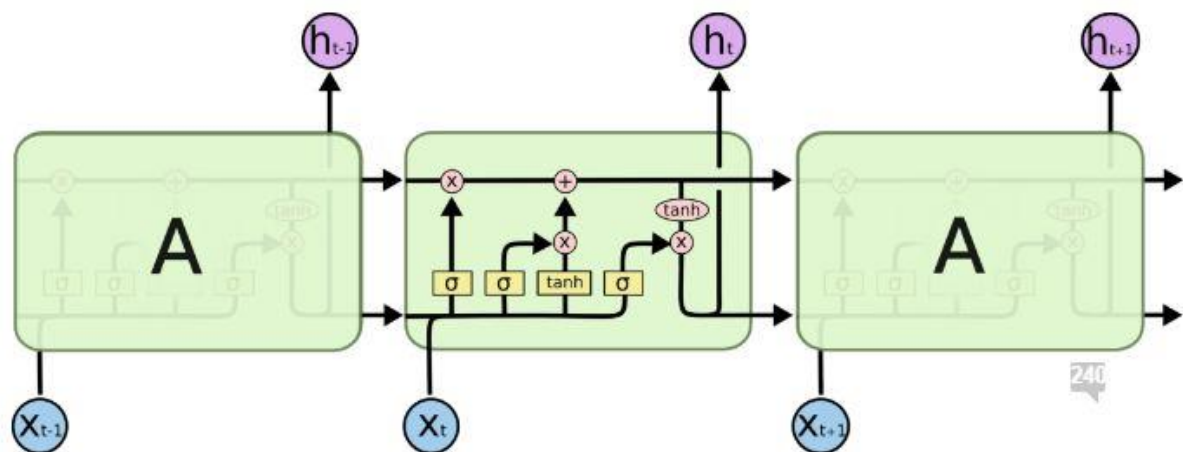
We are passing natural language expression along with the image as an input data which describes an image region to be segmented as explained above. So for the 2nd part of the architecture i.e natural language expression encoder we present a baseline model below. We are proposing per-word segmentation where this model is using “bag-of-word” representation of the expression. In tensorflow, there is a built-in processing tool for determining vocabulary embedding, called “VocabularyProcessor” which takes natural language expressions of all the images and generate document-ID matrix. Then we declare the embedding matrix for the words. Sentence words will be translated into indices. These indices are translated into one-hot-encoded vectors that we have created with an identity matrix, which will be the size of our word embeddings. We use this matrix to look up the sparse vector for each word and add them together for the sparse sentence vector. we have used TensorFlow's embedding lookup function that maps the indices of the words in the sentence to the one-hot-encoded vectors of our identity matrix. Once we have that matrix, we create the sentence vector by summing up the aforementioned word vectors. In short, we have used TensorFlow's vocabulary processing functions to create a standardized vocabulary to work with and created sentence vectors which were the sum of each text's word vectors. Unfortunately we weren't able to get results after concatenating the bag of words embedded matrix with feature descriptors above to the third part of the project, so we just went forward with a better optimized encoder.

2.2 Optimized Algorithm (LSTM):

Since Bag of words is quite a simple model which doesn't capture the relationships between the words so as expected it wasn't much able to perform well. So we move forward and thought to use a model which takes cares of long term dependencies between words and also not have a problem with variable length sequences as our entire Referit data consists of sequences with variable lengths of query expressions. We thought of using RNN for a while, one of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in “the clouds are in the sky,” we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the

relevant information and the place that it's needed is small, RNNs can learn to use the past information.



The repeating module in an LSTM contains four interacting layers.

Fig 4: Internal architecture of a simple variant of an LSTM

But there are also cases where we need more context. Consider trying to predict the last word in the text “I grew up in France... I speak fluent *French*.”

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. The above figure shows a simple variant of LSTM. Since the query expressions were variable in length LSTM sounding like the perfect fit capturing dependencies even in the long expressions .Other benefit of LSTM includes that it can be used as a baseline unsupervised model that helps supervised tasks such as classification and regression later on in the machine learning pipeline and exactly what we are trying to do.

For the input natural language expression that describes an image region, we would like to represent the text sequence as a vector since it is easier to process fixed-length vectors than variable-length sequences. To achieve this goal, we take the encoder approach in sequence to sequence learning methods. In our encoder for the natural language expression, we first embed each word into a vector through a word embedding matrix, and then use a recurrent Long-Short Term Memory (LSTM) network with D-text dimensional hidden state to scan through the embedded word sequence. For a text sequence $S = (w_1...w_T)$ with T words (where w_T is the vector embedding for the t-th word), at each time step t, the LSTM network takes as input the embedded word vector w_T from the word embedding matrix. At the final time step $t = T$ after the LSTM network have seen the whole text sequence, we use the hidden state h_T in LSTM network as the encoded vector representation of the expression. We also L2-normalize the D-text dimensions in h_T . We use a LSTM network with D-text = 1000 dimensional hidden state in our implementation.

3. Fully convolution classification network:

After extracting the spatial feature map from the image in FCN Section and the encoded expression h_T , we want to determine whether or not each spatial location on the feature map belongs the foreground (the visual entities described by the natural language expression). In our model, this is done by a fully convolution classifier over the local image descriptor and the encoded expression. We first tile and concatenate h_T to the local descriptor at each spatial location in the spatial grid to obtain a $w \times h \times D_{main}$ (where $D_{main} = D_{im} + D_{text} + 2$) spatial map containing both visual and linguistic features. Then, we train a two- layer classification network, with a D_{cls} dimensional hidden layer, which takes at input the D_{main} dimensional representation and output a score to indicate whether a spatial location belong to the target image region or not. We use $D_{cls} = 500$ in our implementation.

In order obtain a segmentation mask with higher resolution, we further perform upsampling through deconvolution (swapping the forward and backward pass of convolution operation) [13]. Here we use a $2s \times 2s$ deconvolution filter with stride s (where $s = 32$ for the VGG-16 network architecture we use). The deconvolution operation produces a $W \times H$ high resolution response map that has the same size as the input image, and the values on the high resolution response map represent the confidence of whether a pixel belongs to the target object. The up-sampling idea was inspired from [13].

Training:

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 pixel-wise loss

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

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:

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

Where α_f and α_b are loss weights for foreground and background pixels, In practice, we finnd that training converges faster using higher loss weights for foreground pixels, and we use $\alpha_f = 3$ and $\alpha_b = 1$ in $L(v_{ij}, M_{ij})$, values adopted from [13].

On our dataset, we use the same trainval and test split as in [11,12]. There are 10,000 images for training and validation, and 10,000 images for testing.

The parameters in feature map extraction network are initialized from a VGG-16 network [14] pre-trained on the 1000-class ILSVRC classification task (1.4 million images). We also use knowledge transfer technique to initialize parameters from already trained model from paper [11] where there output was a bounding box rather than a pixel wise segmentation output. Knowledge transfer between same architecture allows us for faster training as well as the chances of optimal value to stick in local minima may also decrease giving better accurate results. The whole network is trained with standard back-propagation using SGD with momentum, also we initialize the filter weights from bilinear interpolation for up-sampling network.

In our implementation, we resize and pad all images and ground-truth segmentation to a fixed size $W \times H$ (where we set $W = H = 512$), keeping their aspect ratio and padding the outside regions with zero, and map the segmentation output back to the original image size to obtain the final segmentation. We evaluate the performance of our model on the 10,000 images in the test set. The following two metrics are used for evaluation: **the overall intersection-over-union (overall IoU) metric** and the **precision** metric. The overall IoU is the total intersection area divided by the total union area, where both intersection area and union area are accumulated over all test samples (each test sample is an image and a referential expression).

Although the overall IoU metric is the standard metric used in PASCAL VOC segmentation [11], our evaluation is slightly different as we would like to measure how accurate the model can segment the foreground region described by the input expression against the background, and the overall IoU metric favors large regions like sky and ground. So we also evaluate with the precision metric at 5 different IoU thresholds from easy to hard: 0.5, 0.6, 0.7, 0.8, 0.9. The precision metric is the percentage of test samples where the IoU between prediction and ground-truth passes the threshold. For example, precision@0.5 is the percentage of expressions where the predicted segmentation overlaps with the ground-truth region by at least 50% IoU.

Results:

The main results for our evaluation are summarized in Table 1. By simply returning the whole image, one already gets 15% overall IoU. This is partially due to the fact that the ReferIt dataset contains some large regions such as "sky" and "city" and the overall IoU metric put more weights on large regions. The main results for our evaluation are summarized in Table 1. By simply returning the whole image, one already gets 15% overall IoU. This is partially due to the fact that the ReferIt dataset contains some large regions such as "sky" and "city" and the overall IoU metric put more weights on large regions. We compare our values with paper [11] and paper [12] whose respective outputs were a bounding box and segments or bounding box output for the latter.

Method	Prec@0.5	Prec@0.6	Prec@0.7	Prec@0.8	Prec@0.9	Total IOU
Our model	33.01%	26.1%	18.31	11.89%	3.92%	47.12%
GroundR [12]	11.08%	6.20%	2.74%	0.78%	0.20%	20.50%
Bounding box[11]	9.73%	4.43%	1.51%	0.27%	0.03%	21.72%

Table 2: The performance of our model and other methods on the ReferIt dataset under precision metric and overall IoU metric

Our model outperforms all the other mentioned methods by a large margin under both precision metric and overall IoU metric. Using the whole bounding box prediction from [11] and [12] achieves comparable precision. As mentioned earlier precision at 0.9 is hardest where intersection and union should almost exactly match for the ground truth and model output scores. Figure 6 shows some failure cases on the ReferIt dataset, where the IoU between prediction and ground-truth segmentation is less than 50%.

In some failure cases (e.g. Figure 6, middle), our model produces reasonable response maps that cover the target regions of the natural language referential expressions, but fails to precisely segment out the boundary of objects. Figure 5 shows some segmentation examples on object regions from our model. It can be seen that our model can predict reasonable segmentation for both object expressions like “bird on the left” and stuff expressions like “sky above the bridge”. Figure 7 visualizes some examples of different referential expressions on the same image.

Code Files:

We implemented whole model using Tensorflow in python, the training was done on Servers(16GB Memory, NVIDIA TESLA K80 GPU with 130GB of disk space) rented from Google cloud services. We have attached the files with a **Readme.txt** , where it is clearly explained how to train the model. There is also attached trained model in the files, which you can use directly to test any image with a query expression to see the result.

Discussion and future work:

In this paper, we address the challenging problem of segmenting natural language expressions, to generate a pixel-wise segmentation output for the image region described by the referential expression. To solve this problem, we propose an end-to-end trainable recurrent convolution neural network model to encode the expression into a vector representation, extract a spatial feature map representation from the image, and output pixel-wise segmentation based on fully convolution classifier. Our model can efficiently predict segmentation output for referential expressions that describe single or multiple objects. Future work includes experimenting with better image descriptors model and encoders for better IOU results.



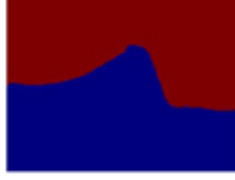








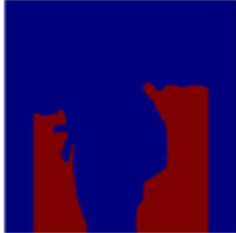
Input Image	Our prediction	Ground-Truth
 <p>Sky above bridge"</p>		
 <p>"Water"</p>		
 <p>"Wall above people"</p>		
 <p>"Ground surrounding her"</p>		

Fig: 5 Shows some segmentation examples on object regions from our model


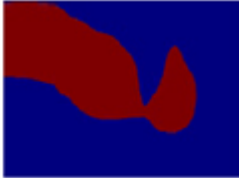




Input Image	Our prediction	Ground-Truth
 <p>"Church"</p>		
 <p>"Right bird"</p>		

Fig: 6 Shows some failure cases on the ReferIt dataset, where the IoU between prediction and ground-truth segmentation is less than 50%.


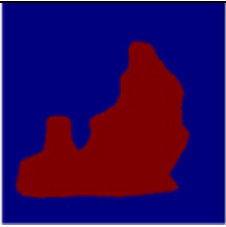



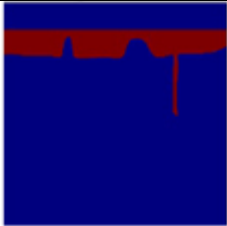

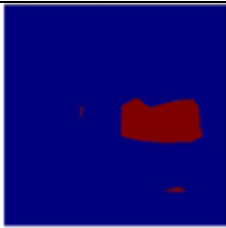
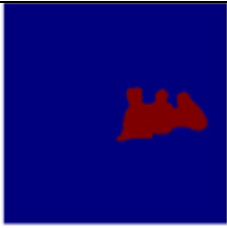

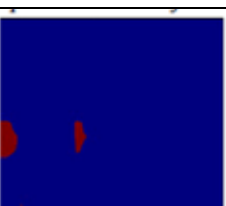
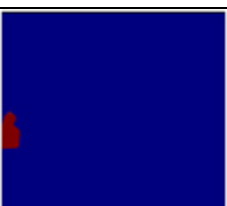
Input Image	Our prediction	Ground-Truth
 <p>"Group of people"</p>		
 <p>"Umbrella"</p>		
 <p>"Three people right"</p>		
 <p>"Person on left"</p>		

Fig: 7 Visualizes some examples of different referential expressions on the same image

Acknowledgements:

We are grateful to Ronghang Hu , EECS Berkeley for providing us with [11] model files, which helped us to initialize our model parameters (Knowledge sharing).

References:

1. Plummer, B., Wang, L., Cervantes, C., Caicedo, J., Hockenmaier, J., Lazebnik, S.: Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). (2015)
2. Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A., Murphy, K.: Generation and comprehension of unambiguous object descriptions. arXiv preprint arXiv:1511.02283 (2015)
3. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 2625-2634
4. Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., Yuille, A.: Deep captioning with multimodal recurrent neural networks (m-rnn). In: Proceedings of the International Conference on Learning Representations. (2015)
5. Kazemzadeh, S., Ordonez, V., Matten, M., Berg, T.L.: Referitgame: Referring to objects in photographs of natural scenes. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). (2014) 787{798
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431{3440
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
8. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random _elds as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1529{1537
9. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1520{1528
10. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015).
11. Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., Darrell, T.: Natural language object retrieval. arXiv preprint arXiv:1511.04164 (2015)
12. Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., Schiele, B.: Grounding of textual phrases in images by reconstruction. arXiv preprint arXiv:1511.03745 (2015)
13. Segmentation from Natural Language Expressions Ronghang Hu, Marcus Rohrbach, Trevor Darrell
fronghang, rohrbach, ,UC Berkeley EECS, CA, United States, ICSI, Berkeley, CA, United States
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)