Referring Image Segmentation via Recurrent Refinement Networks

Ruiyu Li[†], Kaican Li[†], Yi-Chun Kuo[†], Michelle Shu[‡],
Xiaojuan Qi[†], Xiaoyong Shen[§], Jiaya Jia[†,§]

[†]The Chinese University of Hong Kong, [‡]Johns Hopkins University, [§]YouTu Lab, Tencent

{ryli,kcli5,yckuo5,xjqi,leojia}@cse.cuhk.edu.hk, mshu1@jhu.edu, goodshenxy@gmail.com

Abstract

We address the problem of image segmentation from natural language descriptions. Existing deep learning-based methods encode image representations based on the output of the last convolutional layer. One general issue is that the resulting image representation lacks multi-scale semantics, which are key components in advanced segmentation systems. In this paper, we utilize the feature pyramids inherently existing in convolutional neural networks to capture the semantics at different scales. To produce suitable information flow through the path of feature hierarchy, we propose Recurrent Refinement Network (RRN) that takes pyramidal features as input to refine the segmentation mask progressively. Experimental results on four available datasets show that our approach outperforms multiple baselines and state-of-the-art[1].

1. Introduction

Semantic image segmentation is a challenging task that has gained much attention over past years. Although existing segmentation methods [24, 2, 36, 35, 20] become increasingly accurate by incorporating Deep Neural Networks (DNNs) [18, 27, 8, 12] and multi-scale image representations [6, 21], they are constrained to predict a fixed set of pre-defined categories. The limited output space for these methods is insufficient for many real world, language-based applications where targets are represented by natural language expressions.

Recently, there is a surge of interest in combining natural language processing with visual understanding, including image-text retrieval [16, 28, 30], image captioning [13, 29, 32], and visual question answering [1]. Hu *et al*. [9] generalized semantic segmentation to the task of *referring image segmentation*, which contains a broader set of categories represented by natural language expressions.

[1]Code is publicly available at https://github.com/liruiyu/referseg_rrn.

(a) input image (b) segmentation mask for *people* [2]

(c) LSTM-CNN [9] (d) our result

Figure 1. Given an input image (a) and a query "`far right man`", the mask generated by segmentation model (b) is much more precise than the one generated by LSTM-CNN model (c). (d) shows the result of our model, which focuses on the entire entity of interest and has a clear boundary. It is noted that the LSTM-CNN model, which generates the result in (c), uses the same segmentation network as [2].

Referring image segmentation, however, is more challenging than traditional semantic segmentation as it demands a deeper understanding of the image. For example, to answer the language query "`the left two apples on the table`", an algorithm needs not only to distinguish among different instances on the table, but also to localize the two apples on the left.

1.1. Exiting Solutions

Existing methods tackle this problem by first modeling image and language jointly, then conducting segmentation based on the combination of both descriptors [9, 10, 23]. Specifically, Long Short Term Memory (LSTM) networks

encode the given natural language descriptions to vectors, which are followed by Convolutional Neural Network (CNN) to extract spatial feature maps from corresponding images. The two resulting features are concatenated with spatial coordinates [9] or multimodal information [23], and passed through pixel-wise classification networks to produce corresponding masks matching language expressions.

These approaches can be interpreted as generating the foreground masks by looking at a particular combination of features extracted from images and descriptions. Although these methods yield notable improvements, the generated image masks sometimes still fail to capture whole entities. Moreover, they often produce rougher edge estimation compared to those generated from semantic segmentation models [24, 36, 2, 35, 20]. Fig. 1 illustrates that segmentation from combined image and language features is worse than that with image features alone. One reason is that multi-scale information, which is vital in semantic segmentation, is not properly modeled in current systems.

1.2. Our Solution

In this work, we aim to generate higher quality image masks by incorporating multi-scale information to refine segmentation results. We note that a simple multi-scale training and testing strategy is ineffective, since this task requires the whole image region to be processed at once to capture the global information referred by natural language. Thus, we propose Recurrent Refinement Network (RRN), which takes pyramidal features as input and polishes segmentation masks progressively.

Our model consists of two parts. We first use the LSTM-CNN model to encode and fuse image and language representations, which outputs a rough localization of the interested entity. Then we feed the fused representation and pyramidal feature to the recurrent refinement module to refine the mask representation over feature hierarchy. The final output is a pixelwise foreground mask.

Different from the recurrent multi-modal interaction model presented in [23] where the internal mask representation progresses in a word-reading order, our RRN refines segmentation by adaptively selecting and fusing image features at different scales. It mimics the way human solve this problem – first localizing the entity of interest and then drawing segmentation masks progressively.

We evaluate the proposed model on four referring segmentation datasets [14, 25, 33]. We explore different recurrent structures and features that are fed into the refinement module, and show that our recurrent model is the best among them. We also visualize and analyze the internal representation and explain how the result improves over time. Our contribution is threefold.

- We propose RNN for referring image segmentation that refines the segmentation mask progressively.
- We explore different structures of the model and explain our feature hierarchy design.
- Our approach achieves state-of-the-art performance on all the four challenging datasets.

2. Related Work

Referring image segmentation extends semantic segmentation to a wider output space on natural language expressions. In [5], a question-focused semantic segmentation task was proposed that links visual questions and segmentation answers. With the recently collected referring expression datasets [14, 25, 33], Hu *et al*. [9] first approached the task of referring image segmentation. They introduced an end-to-end trainable framework that combines image and language features and outputs pixel-level segmentation of the target region. This model was further improved by utilizing extra large scale vision-only and image-only data [10]. To encode individual interaction between an image and each word, a convolutional multimodal LSTM was proposed to capture multimodal information over time [23].

Different from that of [23], our model focuses on refining segmentation masks recurrently by gradually incorporating multi-scale information. Our work is specifically related to the following areas.

Semantic Segmentation Research on semantic segmentation are mostly driven by advancement of Fully Convolutional Networks (FCN) [24]. FCN converts fully connected layers to fully convolutional ones and uses skip-connection to sum segmentation scores over multiple scales. To remedy the down-sampling issue caused by pooling layers, Chen *et al*. [2] proposed dilated convolution to preserve the spatial resolution while maintaining the size of receptive field of each kernel during convolution. In [20], a multi-path refinement network exploits image details at multiple scales to produce high-resolution segmentation masks. In [35], a pyramid pooling module fuses features under different pyramid scales. The proposed model has won first place in ImageNet Scene Parsing Challenge 2016 [26]. Our method also makes use of pyramid features for segmentation.

Referring Expression Comprehension and Generation Referring image segmentation is closely related to localizing objects in images referred by natural language description. In [11], a framework was proposed to localize the object based on the description reconstruction loss of each bounding box proposal. In [25, 33, 34], the task of referring expression comprehension and generation was generalized to localize the object referred by expression and generate the description for each object simultaneously. Maximum mutual information training generates less ambiguous description for each object [25]. Yu *et al*. [33] modeled vi-

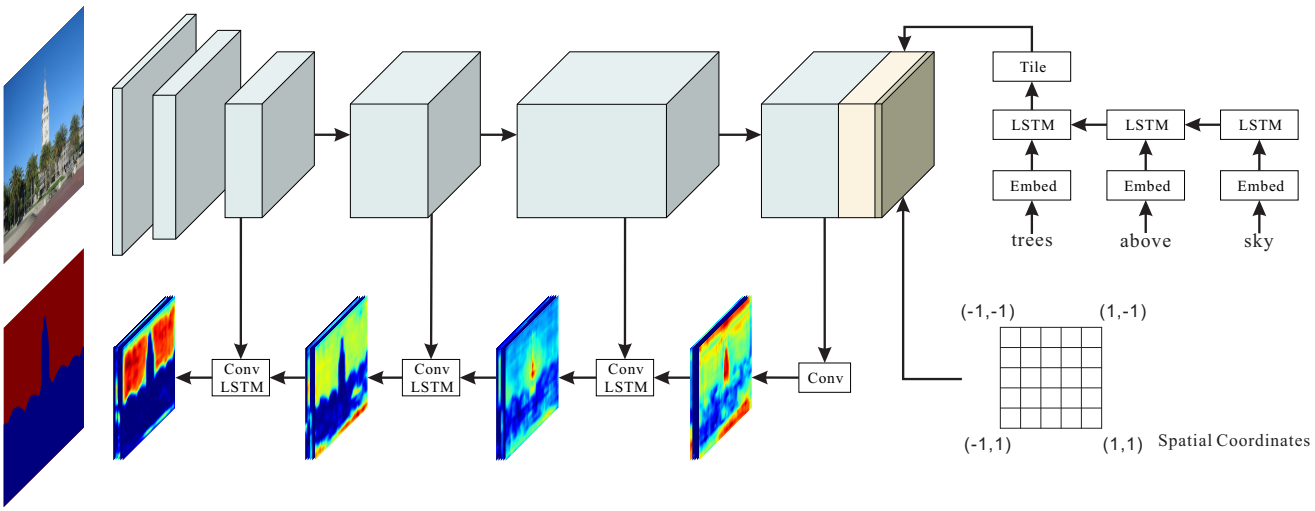


Figure 2. The overall architecture of Recurrent Refinement Network (RRN) designed for referring image segmentation. Here, we adopt DeepLab ResNet-101 [2] to extract image representation. Hence the inputs (conv3, conv4, and conv5 feature maps) of convolutional LSTM are with the same spatial resolution.

sual difference among objects to enhance performance. The method in [34] jointly learns speaker and listener modules via discriminative reinforcer's feedback.

Hierarchical Image Features Feature hierarchy is crucial for building object recognition systems [19, 7, 21]. In [19], the proposed model partitions the image into different sub-regions and calculates the local descriptor within each sub-region. Spatial pyramid pooling layers were used in [7] to pool region features at different scales on the convolutional feature map. In [21], feature pyramids were created via a top-down path with lateral connection. Predictions were made independently at different levels after integrating the multi-scale information. Our method adopts the top-down path similar to that of [21], and yet uses a recurrent mechanism to incorporate the pyramid features.

3. Our Model

The overall architecture of our model is illustrated in Fig. 2. Given an input image and a natural language description as query, our model first uses LSTM and CNN to localize a rough image region of the target indicated by natural language descriptions. Then, we use a recurrent refinement module to generate a high quality pixel-wise segmentation mask. In the following, we begin with our motivation, and elaborate on RRN in Section 3.2. Implementation details are presented in Section 3.3.

3.1. Motivation

The ultimate goal of referring image segmentation is to segment a precise foreground mask corresponding to the queried natural language expression. Previous methods [9, 10, 23] search for a particular correspondence between jointly embedded image and language features. While such correspondences are determined by combining image and language features, the resulting segmentation is generally not as precise as that generated by image feature alone, as shown in Fig. 1.

Another drawback of previous methods is that multi-scale semantics are not sufficiently captured. Multi-scale training and testing is infeasible, since language features are required to interact with image features at every location to ensure correct prediction. For example, to segment a person on the left of image, rescaling and cropping the image may fail to capture the whole interested region. We solve this problem by adopting hierarchical structure to combine different semantic features in a top-down fashion. Different from [21], we devise a gated recurrent regime to aggregate the pyramidal feature, which has the ability to adaptively select image features at different scales controlled by the structures called gates.

3.2. Recurrent Refinement Network

Localization To segment a precise mask queried by a natural language expression, the first step is to localize the corresponding image region. Our localization network is based on the LSTM-CNN model [9]. However, unlike the previous one, which outputs the final segmentation mask directly, we use it to generate a rough mask estimation subsequently refined by the recurrent refinement module.

Given an input image I of size $W \times H$, our model uses a CNN to extract a $w \times h \times D_I$ spatial feature map. Then we perform l_2 normalization on the last dimension of the feature map to obtain a robust representation. To include the spatial information, we concatenate a 8D spatial coordinate

akin to the implementation of [9] at each position to produce a $w \times h \times (D_I + 8)$ image representation.

As for the input natural language description, we encode each word using one-hot vectors and map it to a word embedding space. Then an LSTM takes one word embedding at a time to update its hidden states. The final hidden state s of size D_S is considered as the language representation. Similar to the image feature, we also apply l_2-normalization to the hidden state s.

The language representation s is then concatenated to the image feature at each spatial location to produce a $w \times h \times (D_I + D_S + 8)$ feature map. We use 1×1 convolutional layer to fuse the concatenated features, resulting in a joint representation of image, language, and spatial information. The fused feature q of size $w \times h \times D_Q$ serves as the input to the following recurrent refinement module.

Recurrent Refinement The fused representation generated by the localization module finds a rough region of interest. To obtain a more precise result, we adopt a recurrent mechanism to leverage pyramidal features and refine the segmentation mask.

Given a set of feature maps $m = \{m_1, m_2, ..., m_T\}$ carrying the semantic information at several scales, where T is the number of different feature maps, we first resize them to match the spatial resolution of the fused feature q. Then each of feature maps is fed to a 1×1 convolutional layer followed by non-linear activation function ϕ to match the channel dimension D_Q. The resulting feature maps $\{x_1, x_2, ..., x_T\}$, corresponding to $\{m_1, m_2, ..., m_T\}$, serve as input to the recurrent module.

To refine the fused feature q, we consider it as the initial hidden state h_0 of the recurrent process and let it interact with the features maps $\{x_1, x_2, ..., x_T\}$ sequentially. The tth interaction happens between h_{t-1} and x_t, given by

$$h_t = F(h_{t-1}, x_t; \theta), \tag{1}$$

where F represents the function that outputs updated hidden state h_t by taking the previous hidden state h_{t-1} and current input x_t as input. θ is a set of parameters shared across all ts. In the case with convolutional LSTM [31], updates take the following form.

$$\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i)\,, \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f)\,, \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c)\,, \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o)\,, \\
h_t &= o_t \circ \tanh(c_t)\,,
\end{aligned} \tag{2}$$

where σ denotes the sigmoid function and $\circ$ is the Hadamard product, *i.e.*, element-wise multiplication. c_t is the internal cell state at the tth time step. i_t, f_t, and o_t are the input gate, forget gate, and output gate respectively. These gates regulate the current input information before combining it to the hidden state h_t. W_i, U_i, W_f, U_f, W_c, U_c, W_o, and U_o are parameters of the convolutional operations. b_i, b_f, b_c, and b_o are biases.

After interacting with all the feature maps, the final hidden state h_T has incorporated multi-scale semantics to produce better segmentation results. A convolutional layer is then attached to produce the probability map of foreground p as

$$p = sigmoid(W_p * h_T + b_p). \tag{3}$$

Training Loss During the course of training, we use bilinear interpolation to upsample p to the same size of the input image, and minimize the binary cross-entropy loss function of

$$L = -\frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}(y_{ij}\log(p_{ij}) + (1 - y_{ij})\log(1 - p_{ij})), \tag{4}$$

where y is the binary ground-truth label.

3.3. Implementation Details

Our recurrent refinement module starts with the lowest resolution feature map, which encodes the strong semantic information. We then feed the feature maps to the recurrent refinement module in a descending order of the levels in the pyramid. The last feature map is composed of lower-level information. But its activations are better-localized and hence become essential for precise edge estimation.

The CNN to extract image feature maps is built upon DeepLab ResNet-101 [2] pre-trained on Pascal VOC [4], which reduces the stride of conv4_1 and conv5_1 to 1 and uses dilated convolution to compensate receptive fields. We remove the atrous spatial pyramid pooling module and only use the feature maps of each stage's last residual block to create a feature hierarchy m as in [21]. Input images are resized and zero-padded to $W \times H$. We choose $W = H = 320$ in our experiments, which results in a spatial resolution of 40×40 in the last feature map.

Follow [23], the LSTM for encoding language representation has the maximum length of 20, *i.e.*, we keep the first 20 words for each description. The cell size of LSTM is set to $1,000$. Moreover, both image and language features have dimension $D_I = D_S = 1,000$. The 1×1 convolutional layers that map each m_t to x_t share the same output dimension $D_Q = 500$, which is identical to the size of hidden states in the recurrent refinement module. We choose ϕ to be the rectified activation function (ReLU) $\phi(x) = \max(0, x)$, which, based on our experiments, yields the best performance. We have also conducted experiments on different updating functions F, including convolutional LSTM, vanilla Recurrent Neural Network (RNN), and the

one without sharing weights. Results are presented in Section 4.2.

The network is trained in an end-to-end manner using Adam optimizer [15] with a weight decay of 0.0005. The initial learning rate is set to 0.00025, to which we apply a polynomial decay with power of 0.9. CNN is fixed during training. We choose batch size 1 and stop training after $700K$ iterations.

4. Experiments

4.1. Datasets

We evaluate our model on four available referring image segmentation datasets of ReferIt[14], UNC [33], UNC+ [33], and G-Ref [25].

ReferIt dataset was collected [14] in a two-player game, in which one player clicks on the image region referred by the other player. It contains 130,525 expressions referring to 96,654 segmented image regions in 19,894 natural images. It is built upon IAPR TC-12 dataset [3] and consists of both objects and stuff, *e.g.*, "`water`" and "`ground`". We use the same splits as in [9].

Both UNC and UNC+ [33] are built on top of MS COCO dataset [22] using the two-player game [14]. They only contain object segmentation masks and have an average of 3.9 same-type objects per image. In UNC dataset, no restriction is placed on the referring expressions, but the players in UNC+ are not allowed to use any location word in the description. Hence UNC+ dataset is more complicated than UNC since only appearance information is available when referring to an object. In UNC, there are 142,209 referring expressions for 50,000 objects in 19,994 images, and in UNC+, there are 141,564 referring expressions for 49,856 objects in 19,992 images. We follow [33] to use the same training, validation, and test splits.

G-Ref [25] is also based on MS COCO dataset [22] and contains only object masks. This dataset was collected on Mechanical Turk via independent rounds instead of a two-player game. This collecting strategy results in an average length of 8.43 words in expression, which is much longer than expressions in the other three datasets (with average length less than 4). At least 2 and at most 4 objects of the same type can appear in a single image. It is composed of 104,560 referring expressions for 54,822 objects in 26,711 images. We use the same splits as in [25].

4.2. Results

Setup We follow the setup of [9] and report two metrics of overall intersection-over-union (overall IoU) and precision with different thresholds. Overall IoU calculates the total intersection area between prediction and ground-truth divided by total union area accumulative over test samples. Precision metric measures the percentage of predictions with IoU higher than a pre-defined threshold. We report the results with five different thresholds as in [9].

4.2.1 Ablation Study

We first explore and compare multiple variants in our proposed model, including different updating structures, ways we build the feature hierarchy, and non-linear activation functions used after each update. Table 1 summarizes the results on UNC validation set.

Baseline We re-implement the D-LSTM model [23] with the help of authors' code that is publicly available. We change the kernel size of the last fully convolutional layer to 3×3 and use it as our baseline without recurrent refinement (referred to as D-LSTM-ours). Another difference is that we compute the training loss regarding image resolution instead of feature resolution. Rows 1 and 2 of Table 1 show the improvement by utilizing a larger-size kernel on the last fully convolutional layer and a denser loss function. Our baseline improves the overall IoU by 2.98% and all precision metrics by 0.9-6.8%. The reason of this improvement is twofold. First, a larger size kernel generates finer edge estimate by exploiting the surrounding information. Second, calculating loss regarding image resolution helps estimate a more precise gradient at each position during back-propagation.

Different Feature Pyramids We investigate the effectiveness of incorporating features at different scales in our model. Specifically, we build feature pyramid m described in [21] using feature maps from last residual blocks of different stages. We denote C_X as the feature map from the last output of conv$X$. m=$\{C_5\}$ implies that we only use C_5 to build the feature pyramid. To understand the benefit of using multi-scale information, we choose a simple updating function $F(h_{t-1}, x_t) = a(W_h * h_{t-1} + U_h * x_t + b_h)$, where $a(x) = max(0, x)$ is ReLU activation function. We compare the models that utilize different numbers of feature maps and refer to them as plain structures of our model without sharing weights.

Rows 2 to 7 of Table 1 demonstrate the impact of incorporating features at different scales. We obtain improved performance when we gradually feed C_5, C_4, C_3 features to the network, but it starts to deteriorate when we combine lower-level features (C_2 and C_1). This analysis indicates while the activations in lower-level feature map are better-localized, lack of semantics may cause ambiguity when combining them with higher-level features. Nevertheless, incorporating multi-scale information is beneficial, which can be seen from the comparison between baseline in row 2 and the models utilizing feature pyramids in rows 3-7.

	Model	*prec*@0.5	*prec*@0.6	*prec*@0.7	*prec*@0.8	*prec*@0.9	ovreall IoU
1	D-LSTM [23] (reproduced)	40.57	28.91	18.08	7.14	0.91	43.51
2	D-LSTM-ours	45.49	35.79	24.27	11.93	1.88	46.49
3	Plain, m=$\{C_5\}$	46.82	36.47	25.25	12.25	1.67	46.87
4	Plain, m=$\{C_5, C_4\}$	49.66	39.78	28.36	15.87	3.10	48.56
5	Plain, m=$\{C_5, C_4, C_3\}$	51.84	42.36	30.64	17.39	3.84	49.74
6	Plain, m=$\{C_5, C_4, C_3, C_2\}$	49.99	40.30	29.68	17.21	4.40	48.97
7	Plain, m=$\{C_5, C_4, C_3, C_2, C_1\}$	49.19	38.92	28.12	16.07	3.80	48.30
8	FPN, m=$\{C_5, C_4, C_3\}$	33.83	25.42	16.79	7.94	1.17	39.31
9	RNN, m=$\{C_5, C_4, C_3\}$	49.85	40.22	29.49	16.62	4.05	48.86
10	RNN, m=$\{C_5, C_4, C_3\}$, a=tanh	46.34	37.63	27.56	15.81	3.38	46.35
11	LSTM, m=$\{C_5, C_4, C_3\}$	56.88	47.05	36.05	20.82	4.79	52.48
12	LSTM, m=$\{C_5, C_4, C_3\}$, a=tanh	**60.19**	**50.19**	**38.32**	**23.87**	**5.66**	**54.26**
13	LSTM, m=$\{C_5, C_5, C_5\}$, a=tanh	56.27	45.31	32.67	17.48	2.53	51.41
14	LSTM, m=$\{C_5, C_4, C_5\}$, a=tanh	58.56	48.36	36.23	21.43	4.31	52.81
15	LSTM, m=$\{C_3, C_4, C_5\}$, a=tanh	59.65	49.64	37.00	22.06	4.69	53.81

Table 1. Results of different variants of our proposed model on UNC [33] validation set. We compare different updating structures and pyramidal features fed to our model. m refers to the set of feature maps used to build the feature hierarchy. a=tanh indicates the non-linear function used after each update.

Fig. 3 shows the segmentation masks predicted from models with different feature pyramids. It is obvious that the boundary of right zebra, especially the part between legs, becomes clearer as the model combines more information from different scales. We note that the ground truth annotation misses one leg of the zebra, which is however revealed by our model. Overall, our model with feature pyramid m=$\{C_5, C_4, C_3\}$ (row 5) yields an improvement of 3.25% over the baseline on the overall IoU metric.

Different Updating Functions Given the practice of using feature pyramid m=$\{C_5, C_4, C_3\}$, we then compare performance of employing different updating functions in Table 1 (rows 8-12). FPN refers to the combination strategy used in [21] where the element-wise addition of h_{t-1} and x_t serves as the output. RNN is a recurrent counterpart of the plain structure where the weights of updating function are shared. LSTM refers to the convolutional LSTM with three gates regulating input, output, and internal information.

We first observe that applying the FPN-like connecting module directly results in notable deterioration on all metrics. We believe this is because the original FPN [21] focuses on object detection and generating segmentation proposals. So this simple structure is not applicable to the referring image segmentation task. Rows 5 and 9 show that sharing the weights of the updating function causes slight performance degradation. However, replacing the updating function with more sophisticated LSTM (in row 11) significantly improves the performance by 3.62% on overall IoU metric. The performance gap between LSTM and RNN demonstrates the effectiveness of the gated mechanism when combining different input information. Regularized by these gates, LSTM has the ability to adaptively add or remove features at different scales, and finally retains information useful for generating final segmentation results.

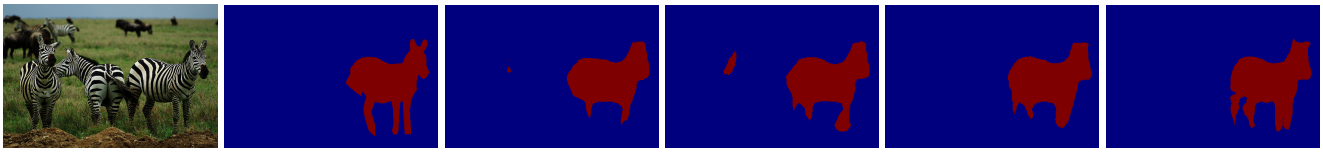
We notice that changing the activation to tanh leads to 1.78% improvement for convolutional LSTM model (in row 12), and 2.51% performance drop for RNN structure (in row 10) on overall IoU metric. We thus use ReLU for RNN and tanh for convolutional LSTM in the rest of our paper.

We have analyzed the convolutional LSTM with feature pyramid m=$\{C_5, C_4, C_3\}$. We now explore setting different input orders of the feature pyramid since the weights of the updating function are shared. Specifically, we compare models with the same number of updates but with different set of pyramidal features.

As expected, using C_5 alone (row 13) leads to a drop of 2.85% on overall IoU metric. Using C_5 and C_4 (row 14) obtains a better result. Yet it is still inferior to the one that utilizes more information at different scales (row 12). These results confirm that our gated recurrent structure indeed benefits from capturing multi-scale information. Finally, we find that the model with the inverse order of feature pyramid (row 15) yields slightly worse performance.

4.2.2 Comparison with State-of-the-arts

We compare the performance of different architectures of our model against state-of-the-arts on the four datasets [14, 33, 25] in Table 2. Here, we only present the overall IoU due to page limit. The full table is presented in the supplementary material. Note that we use the same set of hyper-parameters as explained in Section 3.3 across all the datasets.



(a) input image (b) ground truth (c) baseline (d) m={C_5} (e) m={C_5, C_4} (f) m={C_5, C_4, C_3}

Figure 3. Segmentation masks generated by different models for the query "`zebra right side`". Notice how our model gradually refines the segmentation mask by combining features at different scales.

	ReferIt	UNC			UNC+			G-Ref
	test	val	testA	testB	val	testA	testB	val
LSTM-CNN [9, 10]	48.03	-	-	-	-	-	-	28.14
DeepLab+RMI [23]	57.34	44.33	44.74	44.63	29.91	30.37	29.43	34.40
DeepLab+RMI+DCRF [23]	58.73	45.18	45.69	45.57	29.86	30.48	29.50	34.52
RRN (with plain structure)	60.66	49.74	51.31	49.49	32.73	34.61	29.86	34.43
RRN (with vanilla RNN)	60.86	48.86	49.79	48.68	32.84	34.63	29.96	33.92
RRN (with LSTM)	63.12	54.26	56.21	52.71	39.23	41.68	35.63	36.32
RRN (with LSTM, DCRF)	**63.63**	**55.33**	**57.26**	**53.95**	**39.75**	**42.15**	**36.11**	**36.45**

Table 2. Experimental results of overall IoU metric on different datasets. For all our models, we use feature pyramid m={C_5, C_4, C_3}. DCRF refers to applying DenseCRF [17] to post-process the segmentation results.

Our model incorporating multi-scale information with plain structure already outperforms state-of-the-art by 0.3-5.6% on ReferIt, UNC, and UNC+ datasets. We note that G-Ref is much more complicated than other datasets due to the longer descriptions used for object referral. Hence fusing features over the word sequence in the RMI model [23] yields better performance. Nevertheless, our plain structured model performs on par with the state-of-the-art without applying DenseCRF [17]. This manifests the effectiveness of combining features at different scales.

Employing convolutional LSTM substantially boosts the performance. The most significant improvement is on the testA split of UNC and UNC+ dataset, with absolute improvement of 11.57% and 11.67% respectively. It is because the referring expressions on testA split only contain people, which is easier to segment since our CNN is pretrained on Pascal VOC [4]. The huge performance gap between UNC and UNC+ also demonstrates the importance of location words in the description for this referring image segmentation task. Our overall system after applying DenseCRF achieves the best performance.

4.2.3 Qualitative Analysis

To understand how our model learns to refine the segmentation results over feature hierarchy, we visualize the internal representation of convolutional LSTM after each time it interacts with input features.

Visualization is created by normalizing the strongest activated channel of hidden state and upsampling back to the same size of the input image. Fig. 4 shows the results. We take the first case as an example, given the query expression "`the white wall`", the response first spreads over most of the image (column 3). Then it gradually pinpoints to the queried region after combining new information as illustrated in columns 4 and 5. The last feature map focuses on the region of interest and has a clear boundary, which is essential for producing a precise segmentation mask. This figure illustrates that our model is capable of ruling out irrelevant image regions adaptively.

Fig. 5 presents the segmentation masks generated by different variants of our model on the four datasets. Our model with convolutional LSTM produces the best segmentation results, manifesting the importance of integrating multi-scale information with an appropriate refining scheme. We show more examples in the supplementary material.

5. Conclusion

We have proposed the recurrent refinement networks for referring image segmentation. Our model learns to adaptively incorporate information from a feature pyramid to generate precise segmentation results. We present complete analysis of the variants of our proposed model. On the four available datasets, our model significantly outperforms previous state-of-the-arts. Visualization of internal representation demonstrates the ability of our model to refine the segmentation masks gradually. Future work includes adapting this framework to general semantic segmentation or instance segmentation tasks.

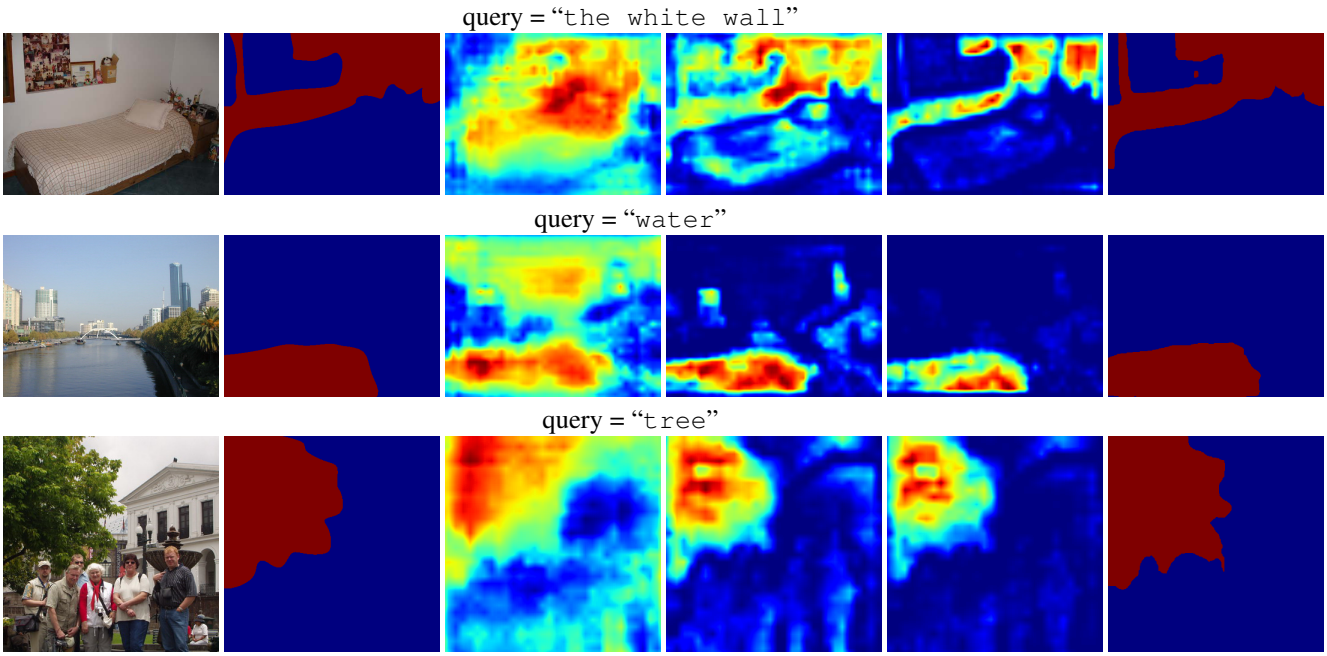


Figure 4. Visualization of the internal representation inside convolutional LSTM. From left to right, we show input images, ground truth masks, the strongest activated channel of hidden states after combining C_5, C_4,C_3 features, and the predicted masks. Our model has the ability to refine the internal representation after each time it interacts with a new input feature.

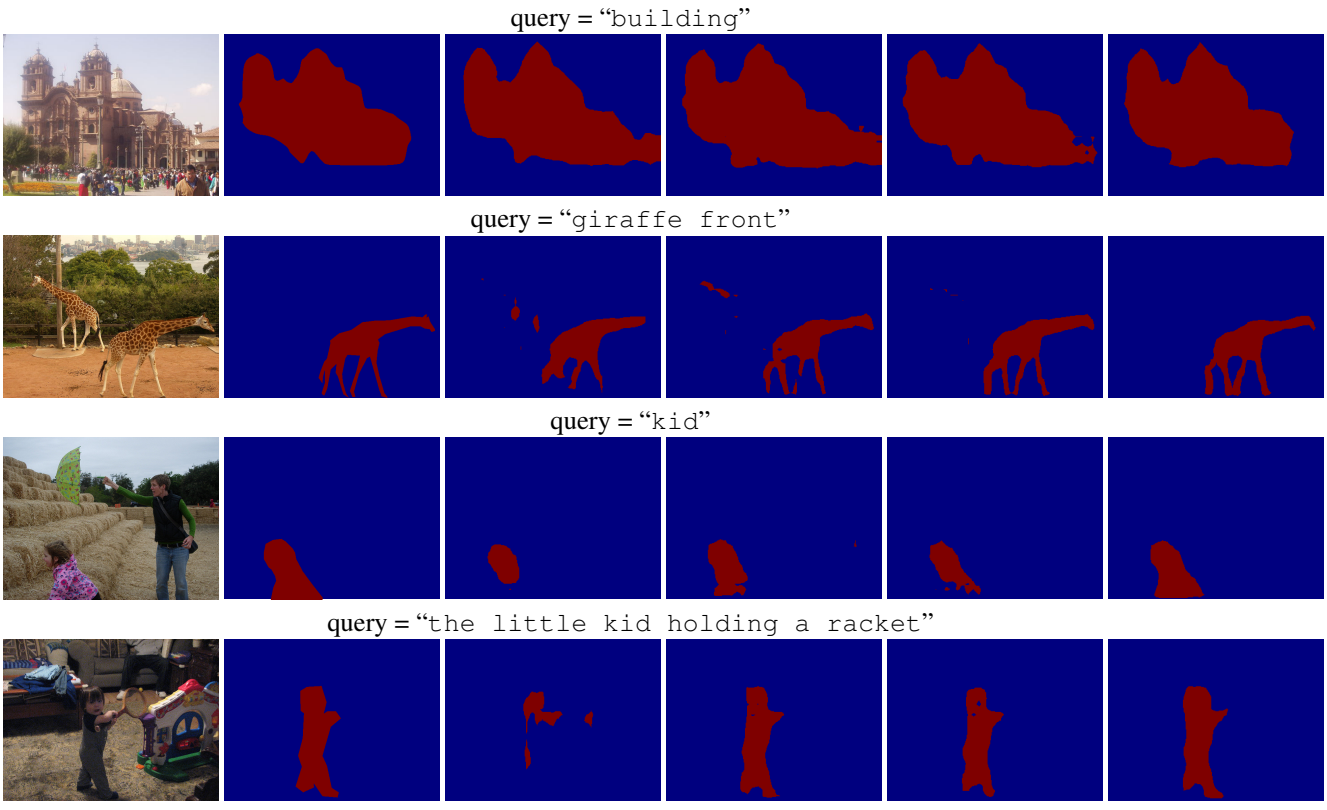


Figure 5. Segmentation results generated by different models. From left to right, we show input images, ground truth masks, and results from baseline, plain structure, RNN, and LSTM respectively. From top to down, we show the samples from ReferIt, UNC, UNC+, and G-Ref datasets.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 1

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 1, 2, 3, 4

[3] H. J. Escalante, C. A. Hernández, J. A. Gonzalez, A. López-López, M. Montes, E. F. Morales, L. E. Sucar, L. Villaseñor, and M. Grubinger. The segmented and annotated iapr tc-12 benchmark. *CVIU*, 2010. 5

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 4, 7

[5] C. Gan, Y. Li, H. Li, C. Sun, and B. Gong. Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation. In *ICCV*, 2017. 2

[6] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 1

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 3

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 1

[9] R. Hu, M. Rohrbach, and T. Darrell. Segmentation from natural language expressions. In *ECCV*, 2016. 1, 2, 3, 4, 5, 7

[10] R. Hu, M. Rohrbach, S. Venugopalan, and T. Darrell. Utilizing large scale vision and text datasets for image segmentation from referring expressions. *arXiv preprint arXiv:1608.08305*, 2016. 1, 2, 3, 7

[11] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *CVPR*, 2016. 2

[12] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1

[13] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *CVPR*, 2015. 1

[14] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referit game: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 2, 5, 6

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[16] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 1

[17] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 7

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 3

[20] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. 2016. 1, 2

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2016. 1, 3, 4, 5, 6

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[23] C. Liu, Z. Lin, X. Shen, J. Yang, X. Lu, and A. Yuille. Recurrent multimodal interaction for referring image segmentation. In *ICCV*, 2017. 1, 2, 3, 4, 5, 6, 7

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2

[25] J. Mao, H. Jonathan, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. *arXiv preprint arXiv:1511.02283*, 2015. 2, 5, 6

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[28] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015. 1

[29] O. Vinyals, A. Toshev, S. Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *CVPR*, 2015. 1

[30] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016. 1

[31] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015. 4

[32] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *ICML*, 2015. 1

[33] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *ECCV*, 2016. 2, 5, 6

[34] L. Yu, H. Tan, M. Bansal, and T. L. Berg. A joint speaker-listener-reinforcer model for referring expressions. *arXiv preprint arXiv:1612.09542*, 2016. 2, 3

[35] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 2

[36] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 1, 2