

An End-to-End Approach to Natural Language Object Retrieval via Context-Aware Deep Reinforcement Learning

Fan Wu^{§*} Zhongwen Xu[†] Yi Yang[†]
[§]Zhejiang University [†]University of Technology Sydney
{jxwufan, zhongwen.s.xu, yee.i.yang}@gmail.com

Abstract

We propose an end-to-end approach to the natural language object retrieval task, which localizes an object within an image according to a natural language description, i.e., referring expression. Previous works divide this problem into two independent stages: first, compute region proposals from the image without the exploration of the language description; second, score the object proposals with regard to the referring expression and choose the top-ranked proposals. The object proposals are generated independently from the referring expression, which makes the proposal generation redundant and even irrelevant to the referred object. In this work, we train an agent with deep reinforcement learning, which learns to move and reshape a bounding box to localize the object according to the referring expression. We incorporate both the spatial and temporal context information into the training procedure. By simultaneously exploiting local visual information, the spatial and temporal context and the referring language *a priori*, the agent selects an appropriate action to take at each time. A special action is defined to indicate when the agent finds the referred object, and terminate the procedure. We evaluate our model on various datasets, and our algorithm significantly outperforms the compared algorithms. Notably, the accuracy improvement of our method over the recent method GroundeR and SCRC on the ReferItGame dataset are 7.67% and 18.25%, respectively.

1. Introduction

Convolutional Neural Networks (ConvNets) has shown phenomenal results [19, 31, 33, 10] for many computer vision applications. With ConvNets, the object detection tasks have been practiced in a more accurate model better than ever. Existing detection algorithms aim to detect a predefined object category from the given image. As a result, de-

*This work was done while F. Wu was visiting University of Technology Sydney.

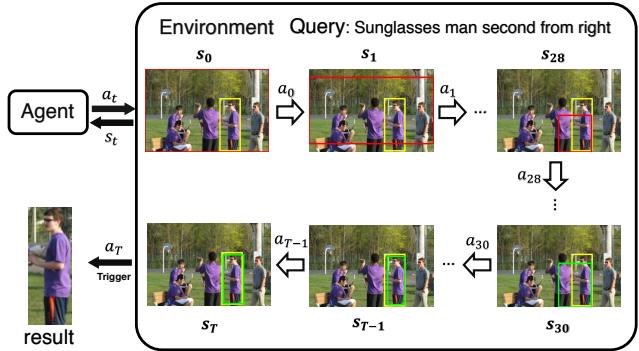


Figure 1. Illustration of the proposed context aware reinforcement learning framework. The yellow box is the ground truth. The bounding box generated by the agent at each time step is green if the Intersection-over-Union (IoU) value with the ground truth box is greater than 0.5, while red otherwise. Best viewed in color.

tection based retrieval systems usually take the target object name as the query, which largely ignore the context information within an image. In the real world, however, the rich information that a user is searching could be more than what a single object name can describe. Compared to object names, language description contains context information such as the relative location of the object, *e.g.*, “the book on your *left-hand side*”, or a specific part of an object, *e.g.*, “*his face*”. With language descriptions, one can even specify detailed attributes of the object of interest, *e.g.* “the man in *middle jeans* and *T-shirt*”. Therefore, natural language provides users with more powerful tools than the scheme of adopting object name as the query.

In this work, we propose a new method of natural language object retrieval. The goal is to localize a referred object in an image or a set of images according to a language description, which can be interpreted as a new type of cross-media retrieval [36]. A typical and straightforward way is to divide the task into two non-overlapping phases. In the first phase, a set of object region proposals are generated as which has been done in [7, 6, 27]. If the algorithm

uses handcrafted features, *e.g.*, EdgeBoxes [41], the quality of proposals may not be good. [39] and [38] use the Fast R-CNN [6] and SSD [21] to generate object detection results as object proposals. However, this type of approaches rely heavily on the training data of object proposals and are restricted to the predefined object categories. As a result, these algorithms [39, 38] can only deal with the pre-defined objects and are not extendable to natural language queries containing new objects and complex reasoning of relative location. In the second phase, these methods adopt a ranking function to locate the region which best matches the description. The limitation of this kind of method is that the two critical phases are conducted independently. In this case, the training process is not well aligned, leading to suboptimal solution for the retrieval task. Furthermore, those approaches usually rely on a large number of proposals to guarantee a satisfactory recall for the target object, which drastically increases redundancies and degrades the discriminative performance of the ranking function.

Inspired by the recent successes of deep reinforcement learning [24, 23, 30], we propose to train a neural network for natural language object retrieval in an end-to-end manner. As illustrated in Figure 1, our method adopts a top-down approach to localize the referred object. Specifically, we define different actions for an agent to change the shape and location of a bounding box. The “agent” takes one of those predefined actions according to the spatial and temporal context, the local image feature as well as the natural language *a priori* at each time step, until an optimal result is reached, *i.e.*, the agent takes a special action (denoted to as a “trigger”) and stops the process. It is worthwhile highlighting the following aspects of the proposed method.

First, our approach performs natural language object retrieval in an end-to-end manner without the pre-computation of proposals, which could be very noisy and redundant. Different from our approach, existing natural object retrieval methods [12, 28, 38] either use handcrafted features or ConvNet features to generate proposals in the first phase. The performance of handcrafted features is comparatively poor. On the other hand, the ConvNet based detectors can only deal with a limited number of predefined object categories. Our end-to-end approach exploits language information and visual information in a joint framework, thereby being able to leverage the mutual benefits of the two inputs for training. Moreover, our approach also avoids the non-trivial task of tuning the number of proposals. Instead, the network decides to stop searching the object by selecting the “trigger” action, thus it constructs a dynamic length search procedure per query.

Second, our approach generates a series of “experiences” to better use the training information under the deep reinforcement learning paradigm [24, 23]. Image-level context information is complementary to local information within

a bounding box [12]. This context is presumably important in natural language object retrieval, especially when the language description contains relative locations. Therefore, we propose to use image-level ConvNet representation as *spatial context*, and explicitly encode such information into the “experience”. Further, a Recurrent Neural Network (RNN) is added into the policy and value networks to track the *temporal context*, *i.e.*, the history states that the agent has encountered. This temporal context would help the agent avoid entering similar mistakes in the previous time steps. The existing approaches [12, 28] merely use the labeled images for training. The “context-aware experience” in our algorithm is generated at each time step after the “agent” takes action. The number of “context-aware experience” is greater than the number of labeled images, and have more diversified information. Meanwhile, as shown in Figure 1, the difference between the bounding boxes at state s_T and state s_{T-1} is subtle but has different IoU values with the ground truth. These subtle differences are also encoded in the “experience”. In that way, our method is able to exploit subtle changes of bounding boxes for a better result.

Third, environment state, agent action and reward function are three key factors for reinforcement learning [32]. Different from a typical deep reinforcement learning scenario, *e.g.*, game playing, computer vision tasks have no well-defined reward function provided by the environment. To address this issue, we define a simple yet effective reward function for the agent. In addition, a potential based reward strategy is also adopted to improve the training speed. Besides, the visual content of an environment are quite similar in game playing scenario. For example, An Atari game [24] has less diversified visual information. Our task is very different because the environment states presented to the agent keep changing dramatically, *i.e.*, natural language queries and images can be very different from one to another. We take advantage of this diversity nature by paralleling a series of agents and environments when collecting experiences in training as practiced in [23].

2. Related Work

Object Detection. Using object proposals to detect object inside image has been validated to be an effective approach. Girshick *et al.* [7] propose R-CNN framework to crop and warp the region proposals generated from off-the-shelf object proposal algorithms, then score each region based on its ConvNet feature. Girshick [6] introduces Fast R-CNN, especially the “RoI pooling” technique to share the feature computation among all the proposal regions, which enhances the processing speed of object detection significantly. Ren *et al.* [27] further improve the object detection system by replacing the external proposal algorithm with a ConvNet which applies sliding windows on the feature maps and outputs bounding boxes. All the methods de-

scribed above are limited to predefined categories, which cannot immediately generalize to other categories.

Deep Reinforcement Learning. Recently, deep reinforcement learning has many breakthroughs. Mnih *et al.* [24] utilize deep neural networks, *i.e.*, Deep Q-learning Network (DQN), to parametrize an action-value function to play Atari games, reaching human-level performance. Silver *et al.* [30] use policy network and value network to play Go and beat a world-class professional player. Mnih *et al.* [23] tackle the training efficiency issue of deep reinforcement learning with an asynchronous approach, making it feasible to train strong agents in a short time on a single machine with CPU only. On the aspect of computer vision applications, Caicedo *et al.* [3] and Jie *et al.* [14] apply the DQN proposed in [24] to generate object proposals in an image with an MDP setting similar to ours. Yeung *et al.* [37] apply a policy gradient method called REINFORCE [35] to detect actions in videos.

Vision and Language. Recurrent neural networks have been widely used in vision and language tasks, starting from image captioning tasks [34]. Recently, Johnson *et al.* [15] propose a model which could be trained end-to-end to localize objects and produce description for dense regions, namely the dense captioning task. Mao *et al.* [22] propose a discriminative training strategy to generate unambiguous descriptions for objects. Yu *et al.* [38, 39] further improve the result on the dense captioning task. Hu *et al.* [12], Nagaraja *et al.* [25] and Rohrbachcite *et al.* [28] focus on retrieving an object inside an image given a language description referring the object. Rohrbach *et al.* [28] use an attention model to localize the language description in an image by choosing the region that could be best used to reconstruct the description. Different from our end-to-end approach, previous works on the natural language object retrieval task *e.g.* [12, 28], use an algorithm to generate object proposals or use the result of an object detection algorithm directly. The referring expression is not utilized in the object proposal or detection procedure.

3. Context-Aware Deep Reinforcement Learning

In this section, we detail the proposed context-aware deep reinforcement learning algorithm.

3.1. Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a sequential decision process, which describes how an agent could interact with an environment and what will happen after each interaction. We denote the MDP as (S, A, R, γ) , where S is a set of states of the environment, A is a set of actions of which the agent could choose from to act on the environment, $R : S \times A \rightarrow \mathbb{R}$ is a reward function that maps a

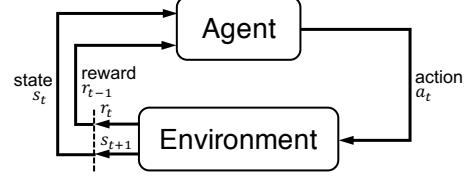


Figure 2. An illustration of the interaction between an agent and an environment [32].

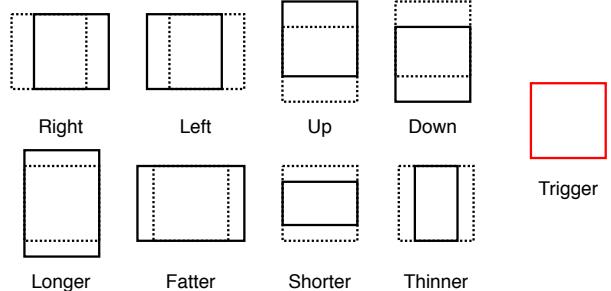


Figure 3. The actions for the agent. The dashed line indicates the bounding box before the action. The solid line is the bounding box after the action. The trigger indicates termination.

state-action pair (s, a) to a reward $r \in \mathbb{R}$, and $\gamma \in (0, 1]$ is a discount factor determining the decay rate in calculating the cumulative discounted reward of the entire trajectory.

The agent interacts with the environment on time step $0, 1, \dots, T$, where T corresponds to termination. From these interactions, a trajectory $\{(s_t, a_t, r_t)\}_{t=0, \dots, T}$ is generated. At each time step $t \in [0, T]$, the agent takes an action $a_t \in A$ based on the current state s_t of the environment. After receiving the action a_t from the agent, the environment transits from state s_t to state s_{t+1} . And the agent receives a reward r_t from the environment. The agent is to maximize the expected cumulative discounted reward $\mathbb{E}[R_t]$ for each of the state s_t , where

$$R_t = \sum_{k=0}^{T-t} \gamma^k r_{k+t}. \quad (1)$$

In our case, the agent changes the size and the position of a bounding box inside the image by using a set of actions to localize an object according to the referring expression. The details are given as follows.

3.2. Environment States and Actions

We define the environment state $s = \{I, Q, \text{bbox}\}$ as a combination of the image I , the referring expression Q and the bounding box $\text{bbox} = [x_0, y_0, x_1, y_1]$ for localizing the target, where (x_0, y_0) and (x_1, y_1) are the top-left and

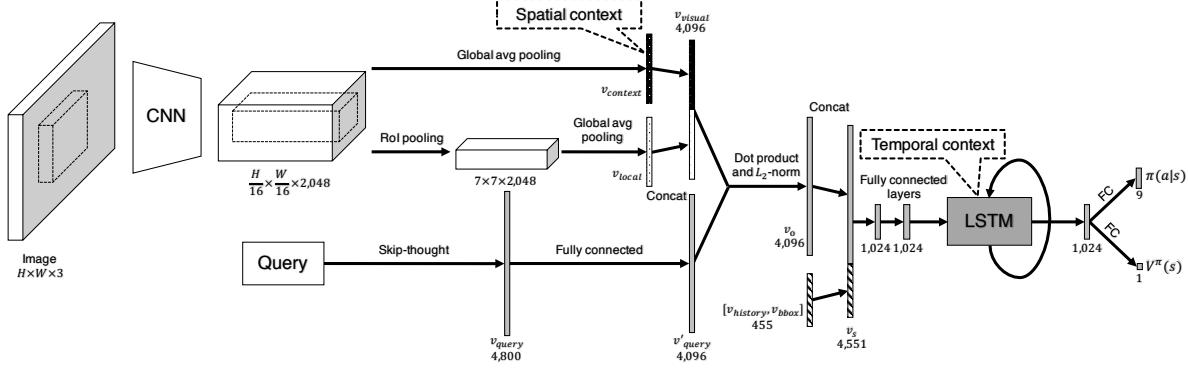


Figure 4. The proposed context-aware policy and value network. The spatial context is computed by applying global average pooling to the entire feature maps. The temporal context is encoded in the state of the LSTM. The outputs of the network are the policy $\pi(a|s)$ and the value $V^\pi(s)$. FC indicates the Fully Connected layer. The numbers under each vector indicate the dimensions of the vector.

bottom-right coordinates respectively. The bounding box is initialized to cover the whole image.

The nine actions as shown in Figure 3 can be categorized into three groups. Four actions of the first group move the location of the bounding box. Another four actions of the second group change the shape of the bounding box. We have an additional action “trigger” to indicate that the agent has achieved an optimal result. Each action moves the top-left point and bottom-right point of the bounding box to adjust it, where the change is proportional to the height and the width of current bounding box. We denote the absolute changes at x and y coordinate directions as $|\Delta x| = \delta \cdot W_{\text{bbox}}$ and $|\Delta y| = \delta \cdot H_{\text{bbox}}$, where W_{bbox} and H_{bbox} are the width and the height of the bounding box respectively. For the movement actions, we set $\delta = 0.2$. For the shape changing actions in the second row, we set the factor $\delta = 0.1$. For example, if the agent takes an “UP” action, the bounding box will be changed from $[x_0, y_0, x_1, y_1]$ to $[x_0, y_0 - 0.2 \cdot (y_1 - y_0), x_1, y_1 - 0.2 \cdot (y_1 - y_0)]$.

3.3. Reward Shaping

In computer vision applications of reinforcement learning, we need to define our reward function, instead of using the reward signals provided by the environment directly. An appropriate reward function for natural language object retrieval task is an essential factor of the success of this work.

For a state-action pair (s, a) , we define our reward function $R(s_t, a_t)$ as follows:

$$R(s_t, a_t) = \begin{cases} R'(s_t, a_t) + F(s_t, a_t) & \text{if } a_t \neq \text{trigger} \\ E(s_t, a_t) & \text{if } a_t = \text{trigger} \end{cases} \quad (2)$$

In the above formulation, $R'(s_t, a_t)$, $F(s_t, a_t)$ and the

$E(s_t, a_t)$ are defined as follows,

$$R'(s_t, a_t) = \begin{cases} \text{IoU}(s_{t+1}) & \text{if } \text{IoU}(s_{t+1}) > \text{IoU}(s) \\ -p & \forall s \in s_0 \dots t \\ & \text{otherwise} \end{cases} \quad (3)$$

$$E(s_t, a_t) = \begin{cases} \eta & \text{if } \text{IoU}(s_t) > \tau \\ -\eta & \text{otherwise} \end{cases} \quad (4)$$

$$F(s_t, a_t) = -\Phi(s_t) + \gamma\Phi(s_{t+1}) \quad (5)$$

$$\Phi(s_t) = \text{IoU}(s_t), \quad (6)$$

In the equations above, s_{t+1} is the state of environment after the agent takes action a_t , η is the quantity of the reward for “trigger”, τ is a threshold of IoU value, and $-p$ is the penalty imposed on the agent when it makes no progress. The IoU function measures the Intersection-over-Union between the current bounding box and the ground truth box of the target in the current state.

The basic reward $R'(a_t, s_t)$ equals to $\text{IoU}(s_{t+1})$ when the new state s_{t+1} has a higher IoU value than all the other states the agent has encountered so far. Otherwise, a penalty $-p$ will be given to the agent. We use $p = 0.05$. Intuitively, this reward function encourages the agent move towards high IoU value states. However, this reward signal is rarely positive. It is hard for the agent to find the goal only with this reward. We add an additional reward, called potential based reward $\Phi(s)$. It is constructed from $\text{IoU}(s)$ function as shown in Eqn (5) and Eqn (6). This kind of reward can accelerate the training process [26]. Lastly, the termination reward function E is decided by the IoU value in the termination state s_T . If $\text{IoU}(s_T) > \tau$, a positive reward η will be generated. Otherwise, the agent will receive a penalty $-\eta$. We set $\tau = 0.5$ and $\eta = 1.0$ empirically. Our discount factor γ is set to 0.99 as in most deep reinforcement learning literatures [24, 23].

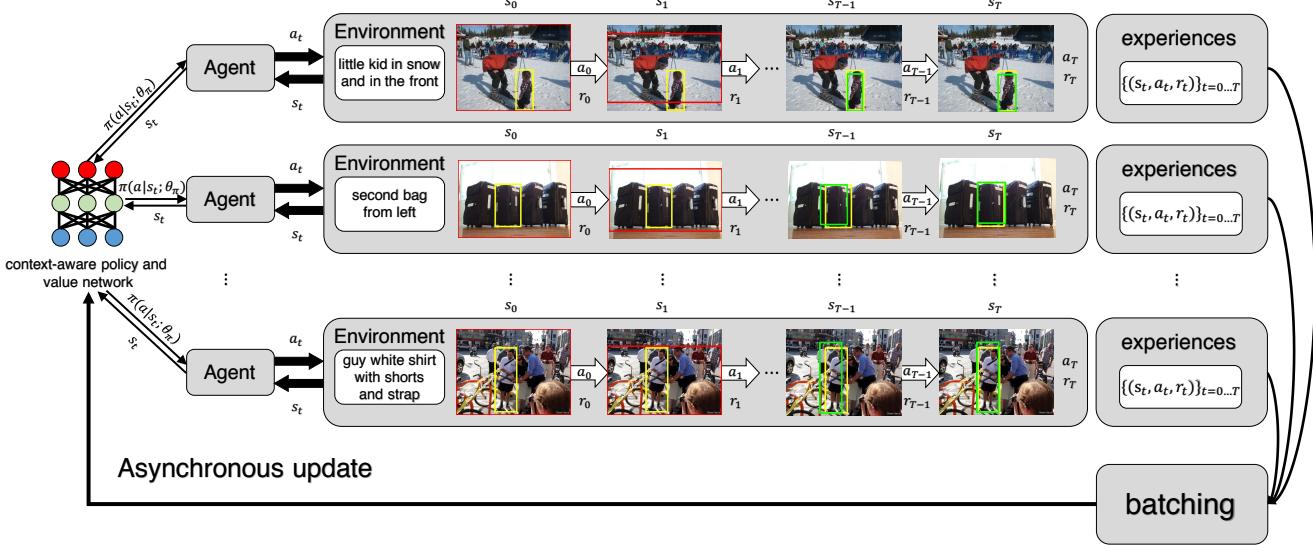


Figure 5. Overview of our training pipeline, we use multiple agents with environments to inference on the current network in parallel. An agent with each query generates a sequence of experiences. Note that the termination time T for each query is variable. A data collector collects the training tuples from all agents, batches the data to update the context-aware policy and value network shown in Figure 4 asynchronously. The color of the bounding box is green if its IoU between the ground truth box is over 0.5, and red otherwise. Best viewed in color.

3.4. Policy and Value Networks

Our agent uses a policy function $\pi(a|s)$ to get a distribution of actions given a state s , and then decides which action to take according to the probabilities over actions. The agent also uses a value function $V^\pi(s) = \mathbb{E}[R_t|s_t = s]$ to estimate the expected cumulative discounted reward R_t from any state s under the policy π . As Figure 4 shows, we use a neural network to parametrize the policy function and value function. These two functions share a common network until the last fully-connected (FC) layer [13, 23]. The network takes the state of the environment as input, and outputs the distribution $\pi(a|s)$ over discrete actions and the value estimation $V^\pi(s)$ of the state s . The ReLU activations are applied between the FC layers.

Our network uses the ResNet-152 [10] which is pre-trained on the ImageNet dataset [29] to extract the visual feature. To encode the spatial context information, we feed the image with width W and height H into a modified ResNet-152 model which has been applied with the atrous algorithm [4] on the conv₅ stage, resulting in image feature maps of size $\frac{H}{16} \times \frac{W}{16} \times 2048$. The feature maps are then fed to a ROI pooling layer [6, 14] to compute the local feature maps inside the bounding box of size $7 \times 7 \times 2048$. We feed these two feature map groups to two global average pooling layers [10] to obtain two visual feature vectors v_{context} and v_{local} . v_{context} is the spatial context, and it is only computed once for all time steps. We denote $v_{\text{visual}} = [v_{\text{context}}, v_{\text{local}}]$. For the language aspect, we uti-

lize skip-thought vectors [18] trained on the BookCorpus dataset [40] to encode the query description. We denote the encoded query feature as v_{query} , which is then projected to $v'_{\text{query}} \in \mathbb{R}^{4,096}$ by a FC layer. After applying dot product and L_2 -norm to v'_{query} and v_{visual} , we obtain the observation of the current state as $v_o = \frac{v'_{\text{query}} \cdot v_{\text{visual}}}{\|v'_{\text{query}} \cdot v_{\text{visual}}\|}$.

However, after the operations above, the computed vector v_o may lose considerable amount of information which is originally in the state s . Thus we propose to leverage the temporal context which tracks the states that the agent has encountered as well as all the actions that the agent has taken. In this paper, 50 previous actions are recorded, which generates a history vector $v_{\text{history}} \in \mathbb{R}^{450}$. Following [22], we define $v_{\text{bbox}} = [\frac{x_0}{W}, \frac{y_0}{H}, \frac{x_1}{W}, \frac{y_1}{H}, \frac{S_{\text{bbox}}}{S_{\text{image}}}]$, where S_{bbox} and S_{image} are the areas of bounding box and image. We use $v_s = [v_o, v_{\text{history}}, v_{\text{bbox}}]$ as the vector representation of state. After passing the v_s to two FC layers with the same output size of 1,024, a Long Short-Term Memory (LSTM) [11] cell with Layer Normalization [2] is used to track the past states [23, 13, 9]. The state inside the LSTM cell is the temporal context for subsequent decision making. Specifically, the output of LSTM will be passed to two FC layers without activation function respectively. One FC layer outputs the policy $\pi(a|s)$ (followed by the softmax operation). The other FC layer outputs the value $V^\pi(s)$.

3.5. Training

An on-policy algorithm [32] interacts with the environment, then uses its own experiences $\{(s, a, r)\}$ to update the current policy. Using a single agent to collect experiences from the environment may get data highly correlated. Updating from such experiences would lead the agent to a suboptimal solution. Therefore, we adopt the asynchronous advantage actor-critic (A3C) method [23] which uses multiple agents associated with environments to collect data in parallel and updates the policy asynchronously.

As Figure 5 shows, we use multiple agents that share a common and global neural network. We denote the policy function and value function from the network as $\pi(a|s; \theta_\pi)$ and $V(s; \theta_v)$, where θ_π is the parameters of the network outputting the policy function, and θ_v are the parameters of the network outputting value function. For one query, an agent uses the current network to interact with the environment constructed by the query. The agent generates an episode $\{(s_t, a_t, r_t)\}_{t=0\dots T}$ for training. After a query is processed by an agent, the agent will randomly select another query to process. The network parameters are asynchronously updated. The actions in an episode may be chosen by different parameters.

Every N consecutive experiences in every episode are grouped. At the time step t , each (s_t, a_t, r_t) is converted to a training tuple (s_t, a_t, R'_t) , where R'_t is defined as:

$$R'_t = \begin{cases} \sum_{k=t}^{t_m(t)-1} \gamma^{k-t} r_k + \gamma^{t_m(t)-t} V(s_{t_m(t)}) & \text{if } t + N \leq T \\ \sum_{k=t}^T \gamma^{k-t} r_k & \text{otherwise} \end{cases} \quad (7)$$

In Eqn (7), $t_m(t) = \lceil \frac{t}{N} \rceil \cdot N$. We set $N = 5$ as in [23]. All the tuples are collected in parallel, and used to optimize in batch mode as follows:

$$\theta_\pi \leftarrow \theta_\pi + \alpha((R'_t - V(s_t; \theta_v))\nabla_{\theta_\pi} \log \pi(a_t|s_t; \theta_\pi) + \beta \nabla_{\theta_\pi} H(\pi(\cdot|s_t; \theta_\pi))), \quad (8)$$

$$\theta_v \leftarrow \theta_v - \alpha \nabla_{\theta_v} (R'_t - V(s_t; \theta_v))^2, \quad (9)$$

where α is the learning rate, $H(\pi(\cdot|s_t; \theta_\pi))$ is the entropy of the policy [23], β is a hyper parameter, $(R'_t - V(s_t; \theta_v))\nabla_{\theta_\pi} \log \pi(a_t|s_t; \theta_\pi)$ is policy gradient [35] which gives the direction to update the policy such that the agent gets more rewards.

The network is trained by the ADAM optimizer [17]. We set $\alpha = 10^{-4}$ and $\beta = 10^{-2}$ during the training. The learning rate α is halved once.

4. Experiments

4.1. Running Environment Details

We implement our model with TensorFlow [1] and tensorpack¹, running on one NVIDIA GTX-1080 GPU. At training time, we use 50 processes to run agents with environments and one process to run policy and value network. The agent processes communicate with the network process via IPC interface provided by operating system. We make our code and the trained models publicly available upon acceptance.

4.2. Preprocessing and Testing

We preprocess every image, ground truth box and query text in all dataset. For the images and ground truth boxes, we followed [27] to resize every image such that its shorter side length equals 600 pixels, and the ground truth boxes are resized proportionally. We filter out non-alphanumeric characters in the query and convert the rest of the characters to their lowercase as in common practices.

At test time, we fix the policy and value network, and use a single agent to process each query. To get the result deterministically, given a state s_t of the current environment, the agent uses the network to get the probabilities $\pi(a|s_t; \theta_\pi)$ over actions, then takes the action $a_t = \arg \max_a \pi(a|s_t; \theta_\pi)$ which has the highest probability. The agent stops taking actions at the time step T when it uses the “trigger”. The bounding box bbox inside the s_T is the result of the algorithm for the query.

4.3. Experiments on the ReferItGame Dataset

Following [12] and [28], we first test our framework on the ReferItGame dataset [16]. The ReferItGame dataset contains 20,000 images from ImageCLEF IAPR image retrieval dataset. For each object, the dataset uses a segmentation region to describe its shape and location information. In total, there are 238 object categories in the dataset. Since the objects in the original dataset are localized with pixel-level segmentations instead of bounding boxes, Hu *et al.* [12] converted the segmentation data of each object to a bounding box, then split the whole dataset to two subsets the trainval set and testing set. We use the meta-data and split provided by [12]. The processed dataset contains 59,976 instances in the trainval set and 60,105 in the testing set. We train our model on the trainval set. During testing, we use the trained agent to give a bounding box for each query in the testing set.

As shown in Table 1, our method outperforms all previous approaches. This result proves our end-to-end model could better exploit the connection between the visual data and language a priori.

¹<https://github.com/ppwwyyxx/tensorpack>



Figure 6. Examples from the testing set of the ReferItGame dataset. The yellow box is the ground truth. The color of the bounding box is green if the result is correct, and red otherwise. Best viewed in color.

Method	Accuracy
LRCN [5]	8.59%
CAFFE-7K [8]	10.38%
SCRC [12]	17.93%
GroundeR [28]	28.51%
Ours	36.18%

Table 1. Accuracy on the ReferItGame dataset.

Figure 6 shows some examples of our result on the testing set of the ReferItGame dataset. Given an image with different natural language queries, our algorithm can correctly locate these queried objects even if the object is a part of another object, *e.g.*, a person's hat. In addition, when the query is about multiple objects or very complex, our algorithm still achieves good performance, *e.g.*, when the query is "any one".

4.4. Experiments on The RefCOCO, RefCOCO+ and Google Refexp (RefCOCOg) Datasets

RefCOCO			
Method	Test A	Test B	Validation
SCRC with 10 Proposals [12]	14.58%	18.39%	16.39%
SCRC with 50 Proposals [12]	19.36%	20.92%	19.39%
SCRC with 100 Proposals [12]	18.47%	20.16%	19.02%
SCRC with 200 Proposals [12]	16.46%	18.37%	16.88%
Ours	54.78%	41.58%	48.19%

Table 2. Accuracy on the RefCOCO dataset. We only used the training set of the RefCOCO dataset for training.



Figure 7. Examples from the testing set of the RefCOCO dataset. The yellow box is the ground truth. The color of the bounding box is green if the result is correct, and red otherwise. Best viewed in color.

RefCOCO+			
Method	Test A	Test B	Validation
SCRC with 10 Proposals [12]	12.57%	14.69%	14.46%
SCRC with 50 Proposals [12]	15.51%	14.09%	14.00%
SCRC with 100 Proposals [12]	14.43%	13.25%	13.72%
SCRC with 200 Proposals [12]	13.60%	12.13%	12.46%
Ours	40.39%	22.81%	31.93%

Table 3. Accuracy on the RefCOCO+ dataset. We only used the training set of the RefCOCO+ dataset for training.

RefCOCOg	
Method	Validation
SCRC with 10 Proposals [12]	15.09%
SCRC with 50 Proposals [12]	16.91%
SCRC with 100 Proposals [12]	16.91%
SCRC with 200 Proposals [12]	15.29 %
Ours	29.04%

Table 4. Accuracy on the RefCOCOg dataset. We only used the training set of the RefCOCOg dataset for training.

We validate our model on the RefCOCO dataset, the RefCOCO+ dataset [38] and the Google Refexp Dataset (RefCOCOg) [22] in this section. It is worth noting that the referring expressions in the RefCOCO+ dataset contain no location word. In total, the RefCOCO dataset contains 19,994 images with 142,209 descriptions for 50,000 objects. The RefCOCO+ dataset contains 19,992 images with 141,564 descriptions for 49,856 objects. The RefCOCOg dataset contains 26,711 images with 85,474 descriptions for 54,822 objects. We use the original split provided by each dataset. The RefCOCO and RefCOCO+ datasets split their testing

	RefCOCO			RefCOCO+			RefCOCOg
	Test A	Test B	Validation	Test A	Test B	Validation	Validation
Ours w/o spatial and temporal context	46.89%	35.51%	41.25%	32.01%	15.91%	25.57 %	21.75%
Ours w/o spatial context	49.78%	37.33%	42.90	36.74%	19.55%	29.11%	29.14%
Ours full	54.78%	41.58%	48.19%	40.39%	22.81%	31.93%	29.04%

Table 5. A comparison of the results with and without the context information.

RefCOCO		
Method	Test A	Test B
SCRC with 10 Proposals [12]	16.14%	18.96%
SCRC with 50 Proposals [12]	20.93%	21.22%
SCRC with 100 Proposals [12]	19.85%	20.41%
SCRC with 200 Proposals [12]	18.46 %	18.65%
Ours	59.66%	44.49%

Table 6. Accuracy on the RefCOCO dataset with more training data. In this experiment, we increase the number of training data by using the combination of the training and validation sets of the RefCOCO, RefCOCO+ and RefCOCOg datasets.

RefCOCO+		
Method	Test A	Test B
SCRC with 10 Proposals [12]	13.59%	14.69%
SCRC with 50 Proposals [12]	17.11%	14.77%
SCRC with 100 Proposals [12]	16.29%	14.26%
SCRC with 200 Proposals [12]	14.86%	12.44%
Ours	47.05%	29.09%

Table 7. Accuracy on the RefCOCO+ dataset with more training data. In this experiment, we increase the number of training data by using the combination of the training and validation sets of the RefCOCO+, RefCOCO+ and RefCOCOg datasets.

set to two set TestA and TestB. The images in the TestA set contain multiple people, and the images in the TestB set only contain non-human objects.

We use the SCRC algorithm [12], which does not require extra labeled data for proposal detector training, as our baseline. In addition to the training set used in our algorithm and [12], the authors of [38] and [39] used a large amount of extra training data, *i.e.*, the validation set and trainval set of MSCOCO [20], to pretrain object detectors. Therefore, we did not include the results of [38] and [39] for a fair comparison.

We train our model using the training set of each dataset, and test our model on the testing set and validation set of those three datasets respectively. Specifically, both our algorithm and [12] use the training set of the RefCOCO, the RefCOCO+ and the RefCOCOg as training data. We report the results of the SCRC model using Top-10, Top-50, Top-100 and Top-200 proposals on the testing and validation set of the three datasets. The results are reported in Table 2, Table 3 and Table 4. We can see that our algorithm outperforms SCRC [12] for all settings dramatically.

Figure 7 shows some of our sample outputs from the testing set of the RefCOCO dataset. Results show our algorithm could process queries contain relationships with another object as in the skewer example, queries contain mul-

tiple objects as in the orange example, and queries contain complex attributes as in the person example.

4.5. Ablation Study of Context Information

In this subsection we test the effects of the context, *i.e.*, the spatial context and the temporal context, in the reinforcement learning. Recall that our algorithm uses an LSTM as temporal context for state tracking, and uses the image level ConvNets representation as spatial context. We train two modified versions of our algorithm. The first one does not contain spatial and temporal context. The other model only removes spatial context (image level ConvNets representation) from our method. We denote the model without all context infomration as “Ours w/o spatial and temporal context”. The model with only temporal context is denoted as “ours w/o spatial context”. As Table 5 shows, the ablation reveals both spatial and temporal context plays an important role in the context-aware policy and value network.

4.6. Performance with More Training Data

Taking the RefCOCO series dataset as an example, we show the performance improvement when the number of training data increases. To obtain more training data, we merge the training sets and the validation sets of the RefCOCO dataset, the RefCOCO+ dataset and the RefCOCOg dataset, and name it as the RefCOCOmg trainval dataset means merged dataset. Note that the testing set of the RefCOCOg dataset has not been released. We use the testing set of the RefCOCO+ dataset and the RefCOCO dataset as the testing data. If an image is in both the testing set and the RefCOCOmg trainval set, we will remove it from the trainval set. In total, the RefCOCOmg trainval set contains 29,456 images with 352,511 descriptions. The experiment results are shown in Table 6 and Table 7. Compared to Table 2 and Table 3, We observe that as we have more training data, the performances of our method and SCRC [12] both increase. Nevertheless, our method still dramatically outperforms SCRC [12]. Also, our method benefits more from more training data.

5. Conclusion

In this paper, we present an end-to-end deep reinforcement learning model for the natural language object retrieval task. Unlike previous approaches, our model leverages the context information and exploits the visual infor-

mation and language a priori in a joint framework. Extensive experiments on various dataset demonstrate effectiveness of our model. Since our method does not constrain the query object in predefined categories, our method has great potential to be generalized in real world scenarios.

References

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *OSDI*, 2016. 6
- [2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5
- [3] J. C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 3
- [4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 5
- [5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 7
- [6] R. Girshick. Fast R-CNN. In *CVPR*, 2015. 1, 2, 5
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2
- [8] S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell. Open-vocabulary object retrieval. In *Robotics: science and systems*, volume 2, page 6. Citeseer, 2014. 7
- [9] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI-SDMIA*, 2015. 5
- [10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5
- [11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5
- [12] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *CVPR*, 2016. 2, 3, 6, 7, 8
- [13] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *ICLR*, 2017. 5
- [14] Z. Jie, X. Liang, J. Feng, X. Jin, W. Lu, and S. Yan. Tree-structured reinforcement learning for sequential object localization. In *NIPS*, 2016. 3, 5
- [15] J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016. 3
- [16] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 6
- [17] D. Kingma and J. Ba. ADAM: A method for stochastic optimization. *ICLR*, 2015. 6
- [18] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *NIPS*, 2015. 5
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1
- [20] 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. 8
- [21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 2
- [22] J. Mao, J. Huang, A. Toshev, O. Camburu, A. L. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 3, 5, 7
- [23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. 2, 3, 4, 5, 6
- [24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2, 3, 4
- [25] V. K. Nagaraja, V. I. Morariu, and L. S. Davis. Modeling context between objects for referring expression understanding. In *ECCV*, 2016. 3
- [26] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999. 4
- [27] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 6
- [28] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016. 2, 3, 6, 7
- [29] 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*, 115(3):211–252, 2015. 5
- [30] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 2, 3
- [31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1
- [32] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 2, 3, 6
- [33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1
- [34] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 3
- [35] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 3, 6

- [36] Y. Yang, D. Xu, F. Nie, J. Luo, and Y. Zhuang. Ranking with local regression and global alignment for cross media retrieval. In *MM*. ACM, 2009. [1](#)
- [37] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016. [3](#)
- [38] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *ECCV*, 2016. [2](#), [3](#), [7](#), [8](#)
- [39] L. Yu, H. Tan, M. Bansal, and T. L. Berg. A joint speaker-listener-reinforcer model for referring expressions. In *CVPR*, 2017. [2](#), [3](#), [8](#)
- [40] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015. [5](#)
- [41] C. L. Zitnick and P. Dollár. Edge Boxes: Locating object proposals from edges. In *ECCV*, 2014. [2](#)