Collaborative Deep Reinforcement Learning for Joint Object Search

Xiangyu Kong[1*] Bo Xin[2] Yizhou Wang[1] Gang Hua[2]
[1]Nat'l Eng. Lab. for Video Technology, Cooperative Medianet Innovation Center, Peking University
{kong, Yizhou.Wang}@pku.edu.cn
[2]Microsoft Research {boxin, ganghua}@microsoft.com

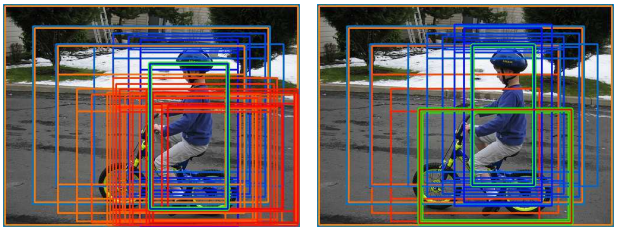
Abstract

We examine the problem of joint top-down active search of multiple objects under interaction, e.g.*, person riding a bicycle, cups held by the table, etc.. Such objects under interaction often can provide contextual cues to each other to facilitate more efficient search. By treating each detector as an agent, we present the first collaborative multi-agent deep reinforcement learning algorithm to learn the optimal policy for joint active object localization, which effectively exploits such beneficial contextual information. We learn inter-agent communication through cross connections with gates between the Q-networks, which is facilitated by a novel multi-agent deep Q-learning algorithm with joint exploitation sampling. We verify our proposed method on multiple object detection benchmarks. Not only does our model help to improve the performance of state-of-the-art active localization models, it also reveals interesting co-detection patterns that are intuitively interpretable.*

1. Introduction

Given an image, the goal of detecting and localizing objects is to place a bounding box around the instances of a pre-defined object class, such as cars, faces, person/people [5, 29, 3, 1]. With the recent advancement [15, 25, 11] of deep convolutional neural networks (CNN) on object classification, generic object detection is also attracting more and more attention with fast increasing detection accuracy on popular benchmarks [8, 22, 21, 17].

Recent detectors explore the idea of bottom-up object region proposals [8], where a relatively small set of a few thousand windows were pre-selected [28] and evaluated. Acceleration were made by sharing computation and pooling over the feature maps from the CNN layers [7, 10]. These works were further accelerated by integrating the separate region proposal step and the classification step into one network [22, 17] by using so-called "anchors" which correspond to regular prototype grid in the image space. However, the number of windows to be evaluated remains several thousand. Therefore, the speed of such region-based methods depends on a heavy use of fast GPUs. When computation power is limited, *e.g.* only CPUs were available, these pipelines are inevitably slow.



(a) Single agent detection (b) Joint agent detection

Figure 1. Joint agent detection compared with single agent detection [2]. The bounding box trajectories are indicated by gradual color change. Blue is for person and red is for bicycle. Successful detections are highlighted in bold green. Both objects were detected within 15 iterations by joint detection while single agent detection failed to locate the bicycle even after 200 iterations. (Only the first 30 iterations were illustrated for visualization purpose).

Active search methods provide a promising complementary top-down scheme to reduce the number of windows to be evaluated [19, 9, 2, 32, 18].When searching or localizing objects, biological vision systems are believed to have a sequential process with changing retinal fixations that gradually accumulate evidence of certainty [14, 16]. It is therefore highly desirable, both biologically and computationally, to explore computational models that facilitate object search in such top-down behavior.

Typically, these models learn policies to search for an object by sequentially translating and/or reshaping the bounding box detector. One can view such a search process as an agent searching for the rewarding ground truth bounding boxes and exploit reinforcement learning (RL) algorithms to learn a good policy. In general, these methods can achieve reasonably good performance using only dozens of steps (effectively the number of windows evaluated).

We examine the problem of joint active search of multiple objects under interaction. On one hand, it is interesting to consider such a collaborative detection "game" played

*Work performed while interning at Microsoft Research.

by multiple agents under an RL setting; on the other hand, it seems especially beneficial in the context of visual object localization where different objects often appear with certain correlated patterns, *e.g.* person riding a bicycle, cups held on top of the table etc. Such objects under interaction often can provide contextual cues to each other [31]. These cues have good potential to facilitate more efficient search policies. We make an initial effort to validate such an hypothesis/intuition by devising a computational model.

We present a collaborative multi-agent deep RL algorithm to learn the optimal policy for joint active object localization. Our proposal follows existing wisdom to exploit RL methods but allows for collaborative behaviors among multiple agents in order to utilize contextual information. In this regard, two key questions are open. i) How to make communications effective in between different agents; and ii) how to jointly learn good policies for all agents.

We propose to learn inter-agent communication through gated cross connections between the Q-networks. This is facilitated by a novel multi-agent deep Q-learning algorithm with joint exploitation sampling and a virtual agent based implementation. Finally, we verify our proposed method on multiple object detection benchmarks. Our model helps to improve the performance of state-of-the-art active localization models and it also reveals interesting co-detection patterns that are intuitively interpretable.

In Section 2, we discuss literatures related to our work. In Section 3, we present the details of the proposed cross Q-network structure and a novel multi-agent deep Q-learning algorithm that effectively facilitate training of the crossed Q-networks. In Section 4, we present comprehensive experiments on multiple popular benchmarks. Section 5 concludes this paper. Here, we summarize our major contributions as follows.

- To our best knowledge, this work presents the first collaborative deep RL solution for joint active object localization.
- We propose a novel multi-agent Q-learning solution that facilitates learnable inter-agent communication with gated cross connections between the Q-networks.
- Our proposal effectively exploits beneficial contextual information between related objects and consistently improve the performance of state-of-the-art active localization models.

2. Related Work

Active search. The idea of active search for localization is not brand new. To name a few, "saccade and fixate" biological pattern were explored in the field of visual attention [14, 16, 30]. In [4], Dollar et al. proposed to estimate pose through cascaded regression steps learnt through gradient descent etc. Latest works on object localization managed to exploit the power of deep learning and achieved more competitive results [19, 9, 2, 32, 18].

In [19], Mnih et al. proposed a recurrent neural network (RNN) based localization network that accumulatively finds numbers from the cluttered translated MNIST dataset. In [9], Garcia et al. proposed to explore statistical relations between consecutive windows and based their model on R-CNN [8] for generic object detection. In [32], Yoo et al. proposed "AttentionNet" where at each current window, a CNN was trained to predict quantized weak directions for the next step to simulate a gradual attention shift. In [2, 18], the authors explicitly deployed deep RL and achieved promising performance with much fewer window evaluations than main stream region proposal methods.

However, none of these works examine the problem of joint active search of multiple objects. In order to exploit beneficial contextual information among differnt objects, we present collaborative multi-agent deep RL. We instantiate our idea with Caicedo and Lazebnik [2] as a single active search model baseline, but our mechanism could be applied to other baseline models with minor adaptation.

Deep reinforcement learning. Recently, the field of reinforcement learning revives with the power of deep learning [20, 24]. Equipped with effective ideas such as experience replay etc., conventional methods, *e.g.* Q-learning, work out very effectively in learning good policies without intermediate supervision for challenging tasks. Our model benefits from these effective ideas in a similar way as recent active methods [2, 18] but with specific novel designs motivated by the joint search problem of interest.

Multi-agent machine learning and reinforcement learning are not new topics. However, conventional collaborative RL methods mostly explore hand-crafted communication protocols [27, 23]. During the preparation of this work, we realize two interesting work that proposed to facilitate learnable communication protocols for multi-agent deep RL [6, 26] and demonstrate superior performance to non-communication counterparts on control management and game related tasks. In [26], Sukhbaatar et al. proposed "CommNet" where policy networks are facilitated with learnable communication channels learnt via back-propagation. In [6], Foerster et al. proposed "Differentiable Inter-Agent Learning" to effectively learn communication for deep Q-networks.

Our proposal share the idea of utilizing back-propagation or designing differentiable communication channels but have different cross network structure with gates and a novel joint sampling Q-learning method. Specifically, our cross network structure used explicit gating mechanism to allow a specific agent to be responsible for certain actions. This is motivated by the problem of object search where one agent usually has primary contribution to the policy. Also dif-

ferent from the training of the unfolded RNNs as in [6], where long range back-propagation may be less effective, our joint sampling design facilitates immediate updates of the parameters and could be easily incorporated into the deep Q-learning algorithm by introducing an auxiliary concept of virtual agent implementation.

3. Collaborative RL for Joint Object Search

We start by recalling a state-of-the-art (single agent) RL method for object localization [2].

3.1. Single Agent RL Object Localization

Reinforcement learning provides a formal framework concerned with how agents take actions in an environment so as to maximize some notion of cumulative reward. Formally, RL defines a set of actions A that an agent takes to achieve its goal; a set of states S that represents the agent's understanding/information of the current environment; and a reward function R that helps to learn an optimal policy to guide the agent's actions based on its states.

In [2], the entire image is viewed as the environment. The agent transforms a bounding box according to a set of actions. The goal of the agent is to land a bounding box at the target object's location. Specifically, the set of actions were defined as follows. $\mathcal{A} := \{$*move right, move left, move up, move down, scale bigger, scale smaller, aspect ratio change fatter, aspect ratio change taller, trigger* $\}$. Each action makes a discrete change to the box by a factor relative to its current size. The action *trigger* means that the agent thinks it finds the object.

The state representation is defined as a tuple $s := (o, h)$. o is a feature vector of the observed region (plus some extra margin for context) extracted from a CNN layer, and h is a fixed-size vector of the action history. The concatenation of o and h is fed into a typical Q-network of two fully connected layers. The network outputs a 9-dimensional vector corresponds to nine action choices. In Figure 2, the networks shown in the same color *e.g.* in blue/red provide illustrations of this architecture.

The reward function $R(a, s \to s')$ is defined for an agent when it takes the action a to move from state s to s'.

$$R(a, s \to s') = sign(IoU(b', g) - IoU(b, g)) \quad (1)$$

where $IoU(b, g) = area(b \cap g)/area(b \cup g)$ is the Intersection-over-Union (IoU) between the target object bounding box g and the predicted box b.

With the action set, state set and reward function defined, the authors in [2] directly applied deep Q-learning [20] to learn the optimal policy. More details on setting parameters can be found in [2]. They also proposed an interesting design for setting masks in the image after taking the trigger action. This design allows for effective detection of multiple instances of the same class. Finally, the authors applied a post SVM classifier to all windows in the trajectory to boost performance.

3.2. Collaborative RL for Joint Object Localization

We generalize the single agent RL model for joint object search. The key concepts include gated cross connections between different Q-networks; joint exploitation sampling for generating corresponding training data, and a virtual agent implementation that facilitates easy adaptation to existing deep Q-learning algorithm.

3.2.1 Q-Networks with Gated Cross Connections

Specifically, Q-learning is an RL algorithm used to find an optimal action-selection policy. The Q-function (action-value function) of a policy π is defined as $Q^\pi(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a]$ where the subscribes of t denote the time step. The optimal action-value function obeys the Bellman optimality equation $Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') | s, a]$ where $r = R(a, s \to s')$ is the specific reward by taking action a to move state s to s' and $\gamma \in [0, 1]$ is a discount factor for future returns.

Deep Q-learning [20] uses deep neural networks to represent the Q-function, *i.e.* $Q(s, a; \theta)$ where θ is the network parameters. (A common choice of the Q-network consists of two fully connected layers as illustrated in Figure 2.) Note that, suppose for each agent i we instantiate one Q-network $Q^{(i)}(a^{(i)}, s^{(i)}; \theta^{(i)})$, in the setting of multi-agent RL, one would naturally desire a Q-function (with a slight abuse of notation, we keep using Q-function here) that facilitates inter-agent communication $Q^{(i)}(a^{(i)}, m^{(i)}, s^{(i)}, m^{(-i)}; \theta^{(i)})$ where $m^{(i)}$ denotes some form of messages sent out from agent i and $m^{(-i)}$ denotes messages received from other agents.

Conventionally, m is often hand crafted based on prior knowledge about the actions and the states. This can be formalized as a function of $m(a, s; \theta_m)$ where θ_m is manually designed. Therefore, a natural idea would be to construct differential messages where θ_m could be learned via gradient back-propagation. This idea is intuitive and reasonable in the same sense of many deep learning successes where learnable features outperform hand crafted ones.

Specifically, we define an agent-wise Q-function as

$$Q := Q^{(i)}(a^{(i)}, m^{(i)}, s^{(i)}, m^{(-i)}; \theta_a^{(i)}, \theta_m^{(i)}), \quad (2)$$

where θ_a and θ_m represents parameters related to actions and messages respectively.

We would now argue that when Q-function were parameterized with deep networks, there are intuitively to the order of L^2 (L is the number of layers of the Q-network) possible configurations for us to construct message channels. This is because the messages could be emitted and received at any layers. Moveover, there should be no global optimal

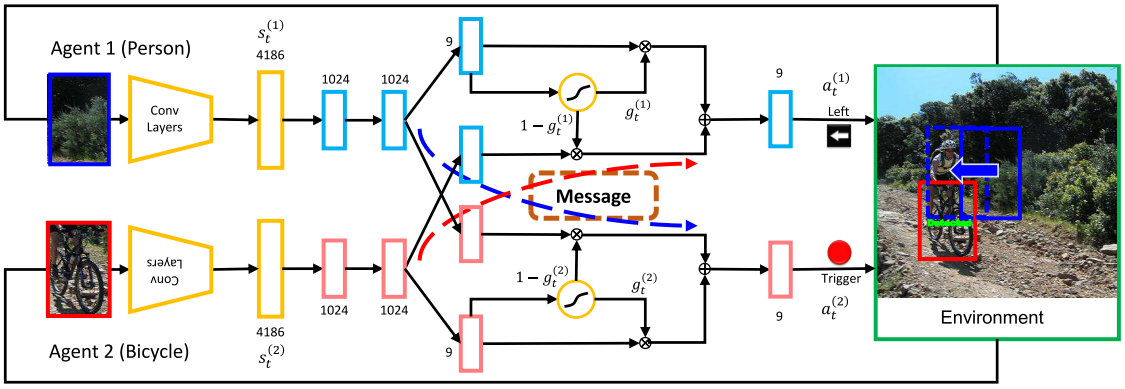


Figure 2. Joint Q-network with gated cross connections and the collaborative reinforcement learning pipeline.

configuration, instead suitable configuration of the message channel should be selected in a problem-dependent manner.

We notice two recent work also propose to facilitate learnable communication protocols for multi-agent deep RL [6, 26] applied to control management and game related tasks respectively. However, we notice that one important insight is missing from the current trend. Messages are often taken-in in a non-discriminative manner and merged with the information flows in the network directly. Actually, allowing the messages to go through an explicit learnable gate (as did in LSTM cells [12]) helps better merging the information and facilitates agent-responsible actions.

The idea is motivated from the search problem of our interest. In general, when searching for a specific object, we would like the agent in charge of detecting the target class to be a primary source of making decisions. Meanwhile we also want to allow other agents to contribute their advices especially when the primary source feels confused in certain situations. Learnable gating mechanism is a natural fit.

Specifically, we design our cross Q-network message channels as illustrated in Figure 2. We add cross connection from the penultimate layer between Q-networks of different agents. We denote the output from this layer of the Q-network of agent i as $\boldsymbol{x}_{L-1}^{(i)}$. We then have

$$\begin{aligned} \bar{\boldsymbol{x}}^{(i)} &= \sigma(\boldsymbol{W}^{(ii)}\boldsymbol{x}_{L-1}^{(i)} + \boldsymbol{b}^{(ii)}) \\ g^{(i)} &= \sigma(\boldsymbol{W}^{(ig)}\bar{\boldsymbol{x}}^{(i)} + \boldsymbol{b}^{(ig)}) \\ \boldsymbol{m}^{(i)} &= \sigma(\boldsymbol{W}^{(im)}\boldsymbol{x}_{L-1}^{(i)} + \boldsymbol{b}^{(im)}) \end{aligned} \tag{3}$$

where σ represent the sigmoid function such that $\sigma(z) = 1/(1+\exp(-z))$.

Now instead of directly inputting $\boldsymbol{x}_{L-1}^{(i)}$ to the next layer as in the single agent case, we also take in the messages from other sources weighted by gates and define

$$\boldsymbol{x}_{L}^{(i)} = g^{(i)} \cdot \bar{\boldsymbol{x}}^{(i)} + (1 - g^{(i)}) \cdot \boldsymbol{m}^{(-i)} \tag{4}$$

Note that, the sigmoid function tends to push the output to approximately 0 or 1. Therefore, with this simple gating induced, we are able to learn effective agent-responsible decisions. This helps us to better understand the searching process. Moreover, now that many actions were effectively determined by one primary agent (and so will the corresponding gradient updates discussed later), one can directly apply learnt networks even when other agents do not co-exist.

3.2.2 Joint Exploitation Sampling

We now turn to the problem of jointly training all Q-networks. Since we do not have any immediate supervision in an RL setting, we cannot directly back propagate gradients in a multi-task manner. The key idea is to jointly sample the next steps during the exploitation phase.

Specifically, in the case of a single agent, in order to reach the Bellman optimality, the Q-learning algorithm proceeds in an iterative fashion. At each iteration, one would sample/choose an action a_t according to the current estimate of the Q-function. One then executes this action a_t in the emulator and observes reward r_t and state s_{t+1}. After this, one updates the parameters of the Q-function by minimizing the distance of $(Q(a_t, s_t; \theta) - (r_t + \gamma \max_{a'} Q(a', s_{t+1}; \theta^-)))^2$. Here θ^- are the parameters of a target network. θ^- can be a copy of the online network but often is another network frozen for a number of iterations while one updates the online network $Q(a, s; \theta)$ [20].

In the multi-agent setting, we propose to sample the action $a_t^{(i)}$ of agent i according to both the activations of itself and the messages from other agents. We jointly perform such sampling to all the agents. For instance, in Figure 2, this corresponds to a joint feed-forward pass from both networks. These samples are later used to update all parameters by jointly minimizing the following distance for all i.

$$\begin{aligned} L^{(i)} := (Q^{(i)}(a_t^{(i)}, m_t^{(i)}, s_t^{(i)}, m_t^{(-i)}; \theta_a^{(i)}, \theta_m^{(i)}) - \\ (r_t^{(i)} + \gamma \max_{a'^{(i)}} Q(a'^{(i)}, s_{t+1}^{(i)}; \theta_a^{(i)-}, \theta_m^{(i)-})))^2 \end{aligned} \tag{5}$$

Since the messages are also differential, joint minimization of the above functions will update parameters related to each of the agents as well as all the message channels

in-between. Specifically, the gradient updates of $\theta_a^{(i)}$ comes from the loss of itself *i.e.* $L^{(i)}$, while the gradient updates of $\theta_m^{(i)}$ comes from the loss of other agents *i.e.* $L^{(-i)}$.

Note that, in principle we could view all agents under one global Markov decision process (MDP) assumption and search for an optimality in the joint action space using the regular Q-learning algorithm. The flip side of this choice, however, is a much larger searching space (81 v.s. 18 in the two agent case) that may require combinatorially much more training data and time. In this regard, the proposed joint sampling strategy can be viewed as an upper-bounding approximation to global optimal. However, we observe that this proposal effectively facilitates gradient back-propagation to all the parameters and can jointly learn good policies for all the Q-networks as desired.

3.2.3 Virtual-Agent Implementation of Joint Training

Intuitively the joint sampling idea can be implemented via simultaneously forward and backward passes through all Q-networks. However in practice, we adopted an alternative implementation with a concept of virtual agents. For each Q-network of an object class, we assign an actual agent detector. Meantime, for each cross network connection we assign a what we call virtual agent. The virtual agents share weights of the corresponding layers with the actual agents. Figure 3 illustrates this idea for the example of Figure 2.

There are two major advantages of this implementation. 1) By considering agents in such a separate manner (and share weights afterwards), we can easily incorporate our design to almost all existing RL algorithms. One can simply implement an extra outer for-loop for all agents followed by necessary weight copying steps. 2) More importantly, this also allows each agent, including virtual ones, to maintain its own pool (replay memory [20]) of samples. These samples are used for updating the corresponding parameters. Note that in modern RL algorithms with deep networks, the concept of replay memory pool are extremely important for stabilizing the learning process.

For example, suppose we would like to jointly train person and bicycle detectors. During training, we have images that contain both classes D_{both} and also images that only contain either person D_{person} or bicycles $D_{bicycle}$. Benefit from an agent-wise replay memory as proposed, the actual person and bicycle agents could be effectively trained with data from $D_{both} \cup D_{person}$ and $D_{both} \cup D_{bicycle}$ respectively, while the cross connections (represented by virtual agents) are only trained with data from D_{both} as desired.

Finally, we update the denotation of the Q-functions in the context of the virtual agent implementation as follows.

$$\begin{aligned} &Q_a^{(i)}(a^{(i)}, s^{(i)}; \theta_{share}^{(i)}, \theta_{self}^{(i)}); \\ &Q_v^{(i\to j)}(a^{(i)}, s^{(i)}; \theta_{share}^{(i)}, \theta_{self}^{(i\to j)}). \end{aligned} \tag{6}$$

The main changes from the definition in Equation (2) are to use $\theta_{self}^{(i\to j)}$ to replace the conceptual out-message $\boldsymbol{m}^{(-i)}$ and to use post addition to replace the conceptual in-message $\boldsymbol{m}^{(i)}$. (Note that, as illustrated in Figure 3, we put the gating part inside the Q-function by definition.) Specifically, we summarize the final multi-agent Q-learning algorithm with joint sampling and virtual agent in Algorithm 1. Although the algorithm applies in general cases, we usually consider only two object classes at the same time, therefore the number of virtual agents is very controllable.

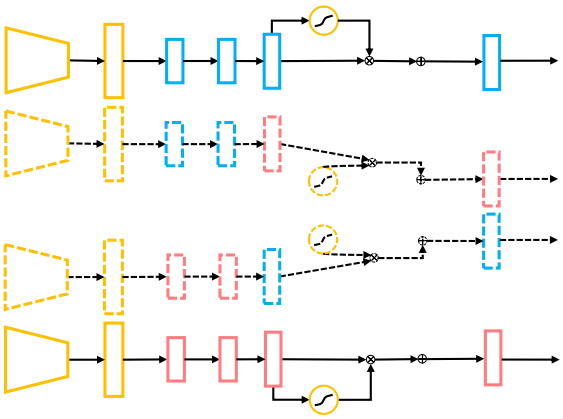


Figure 3. An illustration of the actual and virtual agents of the example in Figure 2. Each row represents one agent and the dashed ones in the middle are virtual agents.

4. Experiments

4.1. Data Construction and Implementation Details

Although different classes of objects co-exist in many situations in real life, there are few datasets explicitly collect data for joint detection tasks. However, we notice that many images from popular detection datasets such as the PASCAL VOC datasets and the COCO dataset have labeled objects of different classes and these images were categorized under all related classes. These images naturally provide a source for us to construct some useful datasets to validate our hypothesis and methods. Specifically, we selected: {*person+bicycle (VOC), ball+racket (COCO), person+handbag (COCO), keyboard+laptop (COCO)*}. With these pairs, we construct two datasets for evaluation purpose. D_1 consists of images that only contain one object for each class. This dataset is used to prove certain concepts since learning and testing tend to be more effective on this relatively cleaned dataset. D_2 consists of all images of the person and bicycle categories from the PASCAL VOC datasets. This one is used to evaluated our proposed method against results of existing single agent models.

For comparison purpose, we implemented the single agent model precisely according to [2]. We manage to have achieved very close performance as reported in [2] though

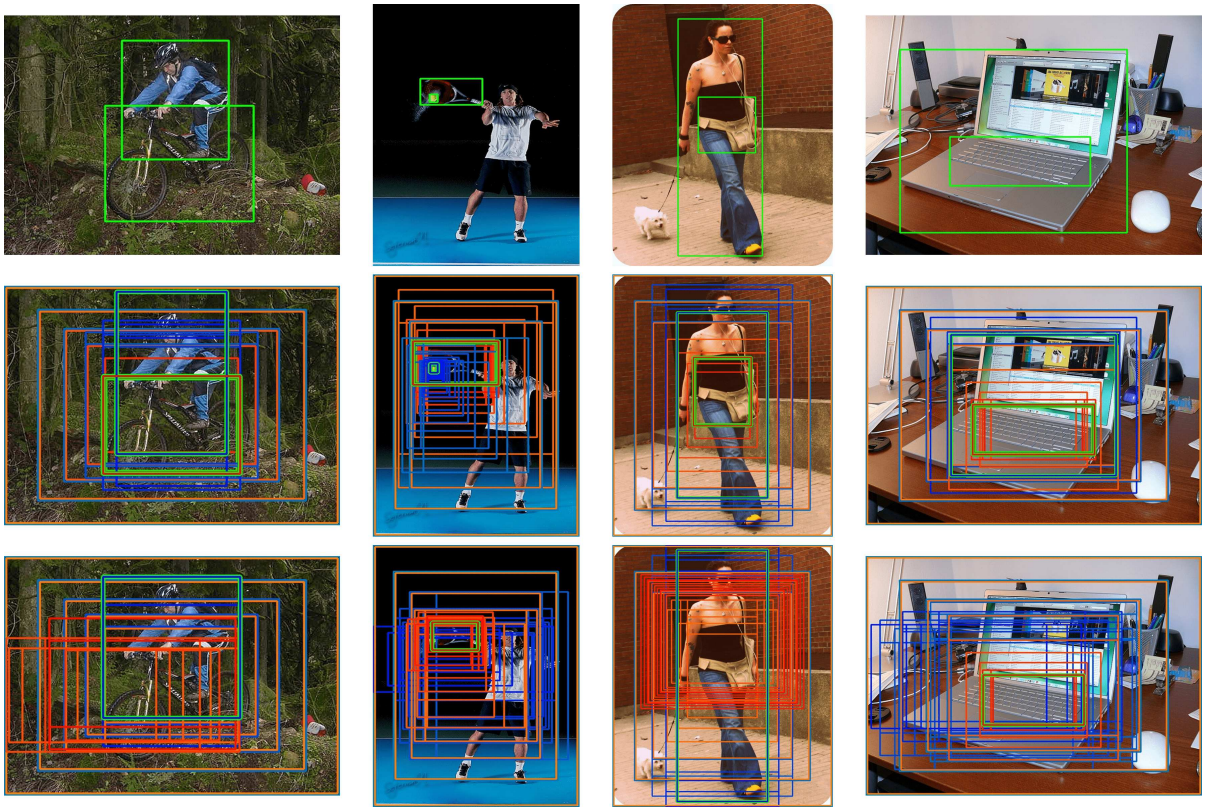


Figure 4. Joint agent detection (mid) compared with single agent detection (bottom). The bounding box trajectories are indicated by gradual color change with blue and red each for one detector. Successful detections are highlighted in bold green.

not exactly the same. The differences may be due to the randomness involved in sampling.

In the case of multiple agents, cross connections between Q-networks are implemented as a fully connected layer from one agent's penultimate layer to another agent's last layer with a post multiplication by a scalar gate as defined in Equation (3). The dimensions are consistent with the corresponding layers in the single agents. For joint training, we initialize each actual single agent network using pre-trained models and initialize cross connection with random weights. We applied the ϵ-greedy strategy of [2] where we have tuned the learning rates to achieve better convergence in our case. We report detection results from the joint model on dataset D_1 since it contains both classes by construction; and report detection results using fine-tuned single agent model by joint training, which demonstrates the ability of the gating mechanism to facilitate agent specific inference and learning.

4.2. Improvement over Single Agent Methods

In Table 3 and Table 2, we demonstrate the performance of our proposal when compared with single agent models. Our joint model consistently outperforms the single agent model on dataset D_1. We notice that on the combinations of *person+bicycle (VOC)* and *laptop+keyboard (COCO)*, the improvement is much more obvious. This is because the configuration of these combinations are relatively more stable across images, *e.g.* person riding a bike and laptop contains the keyboard etc. Meanwhile, the configurations of *person+handbag (COCO)* and *ball+racket (COCO)* have multiple modes in all the images and more "noisy" images that contain little information for co-localization.

When tested on dataset D_2, our joint model also achieved better performance than single active search models. The performance gain is moderate in this case. This is because the number of images containing both object classes is small when compared with that for each category, therefore the extra information gain is diluted. This is especially the case for the person category whose number of images is much larger. Note that, state-of-the-art detection models such as R-CNN [8] and its extensions can achieve better results when using tens of times more windows. But it is not our focus here.

In Figure 4, we illustrate the search process with some examples. In these cases, while our joint detection model successfully locates objects from both categories, the single agent model often only detects one or neither of them correctly. The locations of the final bounding boxes found by the joint model also seem better overlapped with the ground truth objects. Moreover, the number of steps taken by the joint model is much smaller. For example, from top to bottom, the illustrated number of steps for our model are: 10,

Initialize replay memory of all agents $D^{(i)}$;
Initialize all Q-networks with random weights (or potentially with pre-trained networks);
for *episode = 1,M* **do**
  Initialize sequence $s_1^{(i)} = \phi(x_1)$ for all i;
  for *t=1,T* **do**
    With probability ϵ select a random action $a_t^{(i)}$, otherwise select
    $a_t^{(i)} = \arg\max_a \{Q_a(a, s_t^{(i)}; \theta_{share}^{(i)}, \theta_{self}^{(i)}) + \sum_{j \neq i} Q_v(a, s_t^{(j)}; \theta_{share}^{(j)}, \theta_{share}^{(j \to i)})\}$;
    Execute action $a_t^{(i)}$ in emulator and observe reward $r_t^{(i)}$;
    Set $s_{t+1}^{(i)}$ with $s_t^{(i)}, a_t^{(i)}$;
    Store transition $\left(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)}\right)$ in $D^{(i)}$ and $D^{(j \to i)}$ for all j;
    Sample random mini-batch of transitions $\left(s_{t'}^{(i)}, a_{t'}^{(i)}, r_{t'}^{(i)}, s_{t'+1}^{(i)}\right)$ from $D^{(i)}$;
    Set $y_{t'}^{(i)} =$
    $\begin{cases} r_{t'}^{(i)} & if\ terminates\ at\ t'+1 \\ r_{t'}^{(i)} + \gamma \max_{a'} \hat{Q}_a(a_{t'}^{i}, s_{t'+1}^{(i)}; \theta^{(i)-}) & else \end{cases}$;
    Perform a gradient descent step on $(y_{t'}^{(i)} - Q_a(a_{t'}^{(i)}, s_{t'+1}^{(i)}; \theta_{share}^{(i)}, \theta_{self}^{(i)}))^2$ with respect to $\theta_{share}^{(i)}, \theta_{self}^{(i)}$;
    Copy θ_{share}^{i} to all virtual agents $(i \to j)$;
    for $j \neq i$ **do**
      Sample mini-batch from $D^{(j \to i)}$;
      Update $\theta_{share}^{(j)}, \theta_{self}^{(j \to i)}$ of the virtual agents $Q_v^{(j \to i)}$ as above;
      Copy θ_{share}^{j} to actual agent j;
    end
  end
end

Algorithm 1: Multi-agent Q-Learning Algorithm

Table 1. Localization accuracy on D_1. Top: single, bottom: joint.

(VOC)		(COCO)		(COCO)		(COCO)	
person	bike	ball	racket	person	hbag	laptop	kboard
76.9	61.5	52.0	59.3	80.4	45.1	60.6	56.9
86.0	**77.8**	**53.9**	**60.2**	**82.5**	**46.4**	**64.6**	**64.7**

24, 7 and 11 respectively. We show the first 30 steps of the single model for visualization purpose. Actually, in all these three cases, the single agent model failed to locate both objects within 200 steps. In practice, our model only uses several tens of steps to locate both objects and the number of steps are often less than when using two single agents, which was shown to be consistently superior to region proposal methods when using limited number of proposals [2].

Table 2. Localization accuracy on D_2.

D_2	person (VOC)	bicycle (VOC)
Mathe et al. [18]	18.7	31.4
Caicedo et al. [2]	45.7	61.9
Ours (Single)	44.6	62.2
Ours (Joint)	**45.6**	**63.9**
R-CNN [8]	54.2	69.7

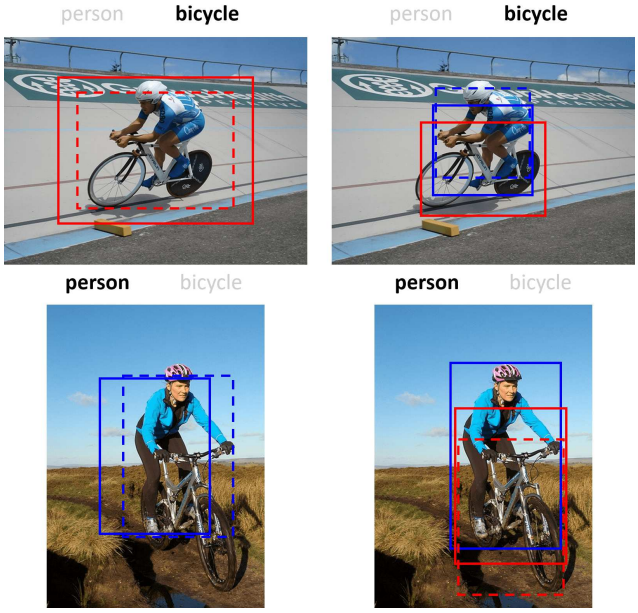


Figure 5. Examples of actions dominated by specific agents. The solid bounding boxes illustrate the current positions of each detector. The dashed bounding boxes illustrate the next positions and indicate the corresponding actions. Blue is for person and red is for bicycle. The agent which dominates the choice of action (by checking the gate value) are highlighted in bold black.

The agents in a joint model help each other in a rational fashion. For example, in the first column of Figure 4, the bicycle looks relatively less distinguishable from the background of bushes. While the single bicycle agent fails to locate its target, in the joint model, the detection of the person seems to help locate the bicycle since it often presents the pattern of a person riding a bicycle. In the second column, the tennis ball looks very small and a single tennis ball agent has trouble finding it; meanwhile benefit from the co-existing pattern with the racket learnt by the joint model, we can successfully detect the ball. The third and fourth column also demonstrate cases where a relatively easy-to-detect object (person and keyboard in these cases) helps to locate the more challenging ones (bag and laptop) due to learnt co-existing patterns.

4.3. Step by Step Examination

In Figure 5, we demonstrate some examples of actions, the choice of which were dominated by specific agents. As the left two images show, when the clue of the primary agent

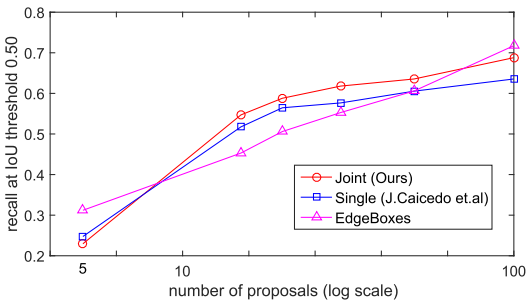


Figure 6. Recall as a function of the number of proposed regions. Compared with region proposal methods, active search methods are better at early recall: only several tens of proposals per image reach 50% recall. Our joint model is even better than the single agent model.

is clear, the actions are often taken according to themselves. For example, given their current input bounding boxes, the bicycle agent knows to scale smaller in the top left image and the person agent knows to move right in the bottom left image.

However, in cases where the primary agent is less confident of itself, our proposal effectively queries information from other agents. For instance, in the bottom right image, the bicycle detector were pushed down, but this action is primarily decided by the person agent. This is probably because the person detector has triggered a target and it feels more certain about the situation. Due to the learnt pattern, it fires relatively strong signals indicating a bicycle underneath and helps to push the red box downwards. Of course this does not necessarily mean the primary bicycle agent has to make the wrong choice of actions, but simply it may be less confident given its relatively noisy current input.

4.4. Evaluation of Recall

Note that, for active search methods, all the regions attended by the agents can be viewed as object proposal candidates. [2] claimed that the single agent localization algorithm can achieve higher recall values when compared with state-of-the-art object proposal methods with limited number of box proposals. We followed their setup and performed the same test to our joint model. In Figure 6, our experiments demonstrate that the proposed multi-agent method has a high recall value when using less proposals. Following the evaluation methods of Hosang et al. [13], we compare the recall of ours with those of the single agent baseline [2] as well as one state-of-the-art object proposal method, Edgebox [33]. The results are from the combination of *person+bicycle (VOC)* which provides stable configurations.

4.5. Failure Case Analysis

In Figure 7, we show one interesting failure case of our method. In this case, our joint model correctly detects the

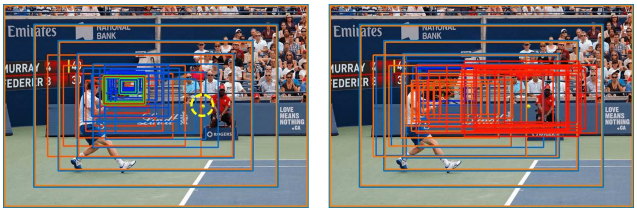


Figure 7. One failure case of joint detection. The true location of the tennis ball is highlighted with dashed yellow circle in the left image. Left: joint agent detection; right: single agent detection.

racket but falsely locates a tennis ball on top of the racket. Meanwhile the true location of the ball is far away to the right. This phenomenon of over-fitting raises one important question. Does joint detection always help? The answer is clearly NO in general cases. Many combinations are not meaningful in the regard of joint detection. Actually, one can barely find shared images for totally unrelated object pairs such as, *e.g.* "bird+car" etc. However, we did explore several more combinations that often coexist but have less spatial correlations. The results are shown as follows.

Table 3. Localization accuracy. Top: single, bottom: joint.

(COCO)		(COCO)		(COCO)		(ImageNet)	
fork	knife	oven	sink	chair	tv	guitar	mike
31.9	45.2	38.2	34.3	35.1	57.1	80.9	45.4
34.7	**46.9**	**42.4**	**37.7**	35.9	56.2	**87.7**	**50.2**

We noticed that even though such pairs do not display a fixed spatial correlation, they often have several major configurations of coexisting patterns. Therefore we can still consistently achieve better performance than single agent models, showcasing that meaningful messages were learned. The pair of "chair+tv" is the least of this case and the positions of chairs and televisions in the images seem rather random even though they often coexist. In this setting, our joint model achieved similar performance with single models. This phenomenon shows that when no clear collaborative information exists, our proposal can perform as well as single agent models without messing up. We attribute this property to the gating mechanism by design.

5. Conclusion

Joint search of multiple objects under interaction often provides contextual cues to each other. By treating each detector as an agent, we present the first collaborative multi-agent deep reinforcement learning method that effectively learns the optimal policy for joint active object localization. Our technical contributions lie in the learnable cross Q-network communications and the joint exploitation sampling strategy. More importantly, we make a first stab to validate the concept of collaborative object localization by devising a computational model, which reveals interesting and intuitive co-detection patterns.

Acknowledgements. GH is partly supported by NSFC Grant 61629301. This work was supported in part by 973-2015CB351800, NSFC-61625201, NSFC-61527804 and the Microsoft Research Asia collaborative research project. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Z GPU used for this research.

References

[1] Shivani Agarwal, Aatif Awan, and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE transactions on pattern analysis and machine intelligence*, 26(11):1475–1490, 2004. 1

[2] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015. 1, 2, 3, 5, 6, 7, 8

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. 1

[4] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1078–1085. IEEE, 2010. 2

[5] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 1

[6] Jakob N Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016. 2, 3, 4

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 2, 6, 7

[9] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3022–3031. IEEE, 2015. 1, 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[13] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *PAMI*, 2015. 8

[14] Laurent Itti, Geraint Rees, and John K Tsotsos. *Neurobiology of attention*. Academic Press, 2005. 1, 2

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[16] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010. 1, 2

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015. 1

[18] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2894–2902, 2016. 1, 2, 7

[19] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014. 1, 2

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2, 3, 4, 5

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 1

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[23] Howard M Schwartz. *Multi-agent machine learning: A reinforcement approach*. John Wiley & Sons, 2014. 2

[24] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 2

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[26] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*, 2016. 2, 4

[27] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 2

[28] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 1

[29] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 1

[30] Wei Wang, Cheng Chen, Yizhou Wang, Tingting Jiang, Fang Fang, and Yuan Yao. Simulating human saccadic scanpaths on natural images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 441–448. IEEE, 2011. 2

[31] Tianfu Wu, Bo Li, and Song-Chun Zhu. Learning and-or model to represent context and occlusion for car detection and viewpoint estimation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1829–1843, 2016. 2

[32] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2659–2667, 2015. 1, 2

[33] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014. 8