Exploring Diverse Ways To Improve An Agent On Active Object Localization With Deep Reinforcement Learning

Jiawei Lu (jl5999) [1] **Lingyu Zhang (lz2814)** [1] **Xinyi Liu (xl3057)** [1] **Yukai Song(ys3493)** [1] **Zixuan Yan (zy2501)** [1]

Abstract

Object localization is a research area with wide range of application scenarios, which has been extensively explored. However, there is a trend to incorporate deep reinforcement learning recently because it is believed to conform better to natural object searching process by human. Although several previous works have built a concrete basis by defining state, action and reward, there is still much field to explore for higher efficiency and accuracy. In this paper, we proposed several improvement in four aspects, including using advanced CNNs to generate state representation, defining more flexible action spaces, changing reward function to avoid undesired activity in agent and using mask instead cross for multiple objects. All these improvements are tested with extensive experiments and proved its effectiveness. They are also flexible enough to get incorporated into other works to achieve even higher performance.[1]

1. Introduction

Object localization has wide application scenarios, for example, industrial production, navigation, agricultural production and target localization. Traditionally, people have been applying the sliding window at different scales for object localization. However, this approach is very space-consuming and time-consuming because of too many candidates to assess. To solve this problem, various methods based on deep learning and convolutional neural networks are proposed to reduce the number of candidate regions that are generated and need to be evaluated. These methods achieve great results recently.

However, based on the rules humans look at an image and localize the target, there is another way to solve the problem by applying Deep Reinforcement Learning (DRL). Previous researchers have made some attempts(Caicedo & Lazebnik, 2015; Bueno et al., 2017; Jie et al., 2016) in this area. The basic idea of this paper is proposed in (Caicedo & Lazebnik, 2015), which will be elaborated in section 2.2. However, the performance of original work is suboptimal and there are still some fields to explore for achieving higher localization efficiency and accuracy. Since the most important components in RL is state, action and reward, and inhibition-of-return (IoR) mechanism is also essential to find multiple objects, we developed different strategies to improve localization from these four aspects: 1. finding a better state space by exploiting advanced cnn architecture; 2. enlarging the action space; 3. exploring better reward functions; 4. applying a better IoR mechanism.

For state space, the original paper used ZFNet as feature extractor. However, there are some of the most popular state-of-the-art pre-trained networks include Vgg16, DenseNet and MobileNet. It is possible to replace the original feature extraction network with these pre-trained advanced networks, in order to achieve better image feature representation and thus have a more complete state space.

For action space, the original work used a fixed stepsize α for agent, which seems rigid. We manage to apply different α instead fixing the stepsize when localizing. It also aligns with how humans localize objects as humans usually do quick scan first and then optimzie the target precisely. When the current state is far from the target, the agent can use the larger step-size and when the current state is close to the target, the agent can use the smaller step-size.

For reward function, in order to have better localization accuracy as well as decreasing the localization steps, we propose two methods for improvement, the first is to change the reward from 1 to 0 when the agent make some improvement in localizing before the terminal state. This approach helps prevent the agent from wandering before termination. The second is to combine the sign of Intersection-over-Union (IoU) and the real variation of IoU in the reward function. The sign of the variation of IoU helps maintain the sensibility of the reward while the variation value of IoU includes the information of the real progress. We use a parameter β to control the value of the variation of IoU and have done extensive tests to find the optimal β.

Last but not least, original paper decided to add a cross when one object is found to avoid repeated localization.

[1]Source code available in: https://github.com/Alexstrasza98/DRL-Object-Detection

Different from adding a cross symbol, which may disturb the information of the image and negatively influence the detection of other targets, we propose a new IoR approach. We cover the detected region with a black mask with the transparency of 0.8. The mask has a superimposed effect so that the more you detect the region, the less likely you will detect this region again.

The rest of this paper is organized as following: section 2 provides a review of previous works on Object Localization with and without reinforcement learning, especially elaborates the original paper we build our project on; section 3 demonstrates how we design the improved algorithm in all four aspects in detail; section 4 includes the experiments and results we conduct on each corresponding aspect, along with comprehensive analysis; section 5 concludes the paper.

2. Related Work

2.1. Previous Works

Object localization has been successfully implemented with sliding window classifiers. One of the most popular sliding window method is based on HOG templates and SVM classifiers. It has been extensively used to localize objects (Felzenszwalb et al., 2010; Malisiewicz et al., 2011), parts of objects (Endres et al., 2013; Lim et al., 2013), discriminative patches (Singh et al., 2012; Juneja et al., 2013) and even salient components of scenes (Juneja et al., 2013). Since sliding windows are category-specific localization algorithms, they are related to our work and are part of our design. However, sliding windows make an exhaustive search over the location-scale space, which deserves careful consideration and modification.

Recently, the trend for object localization is the generation of category independent object proposals. Hosang et al. (Hosang et al., 2014) provide an in depth analysis of ten object proposal methods, whose goal is to generate the smallest set of candidateregions with the highest possible recall. Compared to sliding windows, substantial acceleration is achieved by reducing the set of candidates. Nonetheless, object detection based on proposals still has thousands of windows for a single image that may contain a few interesting objects.

Some efforts have been made to reduce the number of evaluation areas during the detection process. For instance, Lampertet al. (Hosang et al., 2014) proposed a branch-and-bound algorithm to find high-scoring regions only evaluating a few locations. Recently, Gonzalez-Garcia et al. (Gonzalez-Garcia et al., 2015) proposed an activesearch strategy to accelerate category-specific R-CNN detectors. Another related work is from Divvala et al. (Divvala et al., 2009), which uses context to determine the localization of objects. These methods attempt to optimize localized computing resources, which are related to our approach.

Visual attention models have been investigated with the goal of predicting where an observer is likely to orient the gaze (Borji & Itti, 2012). These models typically identify areas of interest based on a saliency map that aggregates low-level features. They are designed to predict human fixations and evaluate performance with user studies (Torralba et al., 2006), whereas our work is designed to locate objects and assess the performance of it.

The machine learning community has been interested in the attentional capabilities of visual recognition in recent years. Xu etal. (Xu et al., 2015) use a Recurrent Neural Network (RNN) to generate captions for images, using an attention mechanism that explains where the system focused attention to predict words. Mnih et al. (Mnih et al., 2014) and Ba et al. (Ba et al., 2014) also used RNNs to select a sequence of regions that need more attention, which are processed at higher resolution for recognizing multiple characters. These models are trained by Reinforcement Learning as we are. However, a simpler architecture and intuitive actions to transform boxes are used in our work.

2.2. Summary of original paper

(Caicedo & Lazebnik, 2015) proposed an active detection model for localizing objects in scenes. The model is a class-specific active detection model that learns to localize target objects known by the system. The proposed model follows a top-down search strategy, which starts by analyzing the whole scene and then proceeds to narrow down the correct location of objects. This is achieved by performing a series of transformations on a box that initially covers a large area of the image, and that box is eventually reduced to a tight bounding box. The sequence of transformations is decided by an agent that analyzes the content of the currently visible region to select the next best action. Each transformation should keep the object inside the visible region while cutting off as much background as possible. Figure 1 illustrates some of the steps in the dynamic decision process for locating a cow in an image.

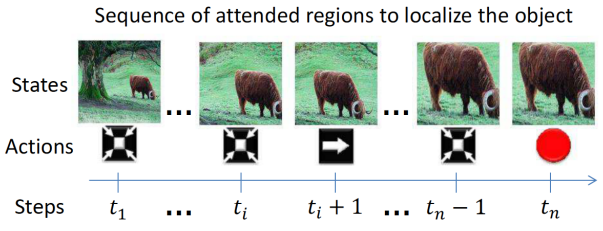


Figure 1. The progress of increasing IoR by iterative analysing current state and taking actions and finally trigger to indicate that an object is found. It is a duplicate figure from the original paper (Caicedo & Lazebnik, 2015).

Their proposed approach is fundamentally different from most localization strategies. Compared with sliding windows, their methods do not follow a fixed path to search objects. Instead, different objects in different scenes end up with different search paths. Unlike object proposal algorithms,candidate regions in their method are selected by high-level reasoning strategies rather than by low-level cues. In addition, compared with the boundary box regression algorithm, their method does not locate objects according to a single, structured prediction method. The proposed dynamic attention-action strategy requires to pay attention to the contents of the current region, and to transform the box in such a way that the target object is progressively more focused.

In this creative work, the entire image is viewed as environment. State is defined as a tuple $s := (o, h)$. Here o is a feature representation of the observed region extracted by a pre-trained CNN; h is a vector of the action history. Actions are defined as a 9-dimensional space $A := \{trigger, right, left, up, down, bigger, smaller, fatter, taller\}$. Each action makes a discrete change to the box by a fixed factor 0.2 relative to its current size. The action trigger means that the agent thinks it finds the object.

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' as following:

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

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

With the action set, state set and reward function defined, the authors directly applied DeepQNetwork algorithm (Mnih et al., 2015) to learn the optimal policy. At the same time, they also put forward a method to set a cross mask in the image after taking the trigger action. This design allows for effective detection of multiple objects.

3. Approach: algorithm development

This section provides detailed description about how we design the improvement for all four aspects. Section 3.1, 3.2, 3.3 and 3.4 discusses state space, action space, reward function and IOR mechanisim respectively.

3.1. State Space

In original paper, a Q-Network takes as input the image inside current bounding box and gives as output the Q value of the nine actions. The whole architecture, as shown in Fig 2, can be divided into two parts. First part is a pre-trained CNN to extract features from raw pixels. It will output a vector of fixed length as o. If we concatenate this o with action history h, we will get the current state that the agent lies in. The definition of h is simple and complete enough, thus we will mainly discuss more possibility of vector o.

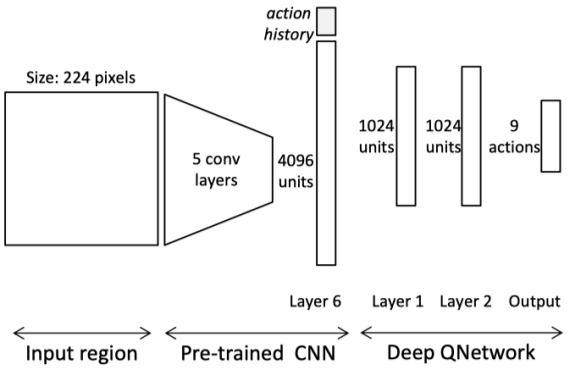


Figure 2. Architecture of the proposed QNetwork. It is a duplicate figure from original paper.

As discussed above, the input of CNN is the image inside current bounding box. This crop of original picture is firstly warped to match the input of the network (224×224) and then put into a convolutional network, consisting of 5 convolutional layers, to extract the feature vector o. The network that originally used is ZFNet (Zeiler & Fergus, 2014), which is put forward in 2013. However, there are much more advanced CNN model which can be used to extract stronger feature vectors. In comparison, we will take Alexnet (Krizhevsky et al., 2012), Vgg16 (Simonyan & Zisserman, 2014), Densenet (Huang et al., 2017), and MobileNet (Howard et al., 2019) to explore how the change in the state space can influence the result.

Alexnet (Krizhevsky et al., 2012) is a milestone in the development of deep learning, which achieved great success in the ImageNet Classification, representing the beginning of the deep learning age. The Alexnet has 60 million parameters and 650, 000 neurons, contains 5 convolutional layers with a final 1000-way softmax layer. The Alexnet is applied to see whether a simple feature extraction network can have a good performance in this work.

Vgg16 from (Simonyan & Zisserman, 2014) first emerged the idea of using blocks. What they did was to increase depth using an architecture with very small (3×3) convolutional layers and push the depth of convolutional layers to 16-19 layers. Since the 16 layer VGG architecture gained the best performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In other improvements, we used vgg16 as benchmark feature extracting network for comparison.

ResNet has achieved (He et al., 2016) unforgettable success in training deeper neural networks through reformulating layers to residual networks, which increased the depth of convolutional networks from 19 in VGG to 152. Densely Connected Networks (DenseNet) further improved the structure of ResNet through several modifications. Similar to Taylor expansion, ResNet decomposes activation function into $f(x) = x + g(x)$, a simple linear term and a more complex nonlinear term, helping each layer of neural network capture information of two layers. In comparison, DenseNet has $L(L+1)/2$ direct connections in the network if the network has L Layers, meaning that each layer connects to every other layer in a feed-forward fashion in the DenseNet. This improvement significantly alleviates the vanishing-gradient problem and encourages feature reuse, improving the result in ImageNet classification greatly. This refinement in the performance became the reason why we trained DenseNet as one of our potential feature extraction models.

The birth of MobileNet represented the start of using deep learning models in embedded devices. In an attempt to enable our model to be implemented in smartphones in the future, we also trained the latest MobileNetV3 (Howard et al., 2019) as feature extraction model. The MobileNetV3 updated its network through two aspects: 1. Layer removal, 1×1 expansion layer, the projection layer, the filtering layer in the last block is removed and 32 filters were reduced to 16 filters 2. Swish non-linearity, $swish(x) = x \cdot sigmoid(x)$ is its mathematical expression. Since a sigmoid function is computationally ineffective, the author modified it to h-swish: $h - swish(x) = x\frac{ReLU6(x+3)}{6}$. These two developments help the MobileNetV3 become faster and more accurate compared with MobileNetV2.

Therefore, if we use these four networks instead of ZFNet as the feature extractor, we believe that a more complete state space can be obtained to improve the overall effect.

3.2. Action space

In original paper, the action space is defined to be a 9 actions set - [trigger, right, left, up, down, bigger, smaller, fatter, taller]. When the agent takes any action except trigger, it will change the position of its bounding box in proportion to current box size. The author fixes this changing proportion α to 0.2 because this value gives a good trade-off between speed and localization accuracy. However, both theoretical analysis and experimental results show that this setup is sometimes rigid. Imagine a situation that there is a small object in the picture. Although the agent perceives that the optimal bounding box is much smaller than current one, it has no choice but to take multiple small steps to zoom in the object. Another situation is that the current bounding box is nearly optimal but needs only a small improvement. The agent also has no choice but to shift the bounding box by 0.2, which may miss some part of the object.

In order to alleviate the problem that each action can only change the fixed size of bounding box, we put forward a new action space for the agent, which makes changing proportion α is optional instead of fixed. Our new action space is a 25 actions sets. The first action is still trigger with no change. The remaining 24 actions can be divided into three groups in order, each group has same 8 action types as original ones but with different α value. We set $\alpha = 0.1$ for first 8 actions, $\alpha = 0.3$ for next 8 and $\alpha = 0.5$ for last 8. The last group can be seen as 'large moves', enabling agent to quickly shrink bounding box to enlarge objects when they are found to be very small. The first group can be seen as 'precise moves', allowing agent to make fine adjustments when the current bounding Box is found very close to objects. And the middle group can be seen as a balance between these two kind of moves. The visualization of new action space is shown in Fig 3.

After defining the new action space, we expect that agent can learn a more flexible way to shrink its bounding box for an object. If it perceives that the target object is much smaller, it should prefer actions with $\alpha = 0.5$ value to approach target quickly. While if it perceives that the target object is very close, it should prefer actions with $\alpha = 0.1$ to optimize the bounding box precisely.

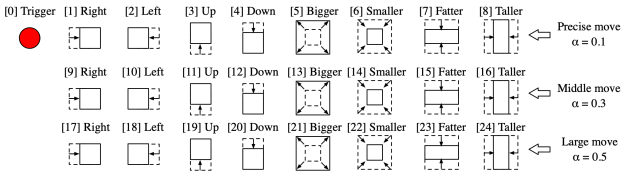


Figure 3. Visualization of 25 actions set. The dotted line represents the bounding box before the action and the solid line represents the bounding box after the action.

3.3. Reward Function

In our algorithm, we proposed two kinds of rewards R, namely $R_a(s, s')$, which represents the reward when the agent chooses action a to move from the current state s to the next state s' when a is not the trigger action, and $R_\omega(s, s')$, which represents the reward when the agent chooses action ω to move from the current state s to the next state s', which is the terminal state, when ω is the trigger action. In the original article, the author proposed the following reward functions:

$$R_a(s, s') = sign(IoU(b', g) - IoU(b, g)) \qquad (2)$$

$$R_\omega(s,s') = \begin{cases} +\eta & \text{if } IoU(b,g) \geq \tau \\ -\eta & \text{otherwise} \end{cases} \quad (3)$$

For non-terminal states, the reward function is $+1$ when the agent have a better performance on localization and is -1 when the localization performance becomes worse. Tthe performance of localization is measured by the Intersection-over-Union (IoU) between the ground truth of the target object and the current region box. Notice that IoU can only be calculated during the training stage because the ground truth is needed. Let b and g be two regions, $IoU(b,g) = area(b \cap g)/area(b \cup g)$. For terminal states, the reward function is $+\eta$ when the IoU is greater than the threshold τ and $-\eta$ otherwise.

Based on the original settings, we discovered two problems and make improvements accordingly.

Improvement 1: Set $R_a = 0$ when making progress

In the original setting, the author set $R_a(s,s') = +1$ when making progress. However, we find that it will result in the agent wandering before the final trigger. To be specific, to achieve a higher reward, the agent will move back and force with little improvement of IoU before getting to the triggering state. It will result in having more moving steps than necessary before triggering. To resolve it, we set $R_a(s,s') = 0$. It can help the agent find a way to the trigger state with less steps. The corresponding rewards for $R_a(s,s')$ is as follows:

$$R_a(s,s') = \begin{cases} 0 & \Delta_{IoU} \geq 0 \\ -1 & \Delta_{IoU} \leq 0 \end{cases} \quad (4)$$

Improvement 2: Add IoU information in R_a

The original setting only considered the sign of the variation of IoU, in our improved version, we will also consider the variation of IoU. The corresponding rewards for $R_a(s,s')$ is as follows:

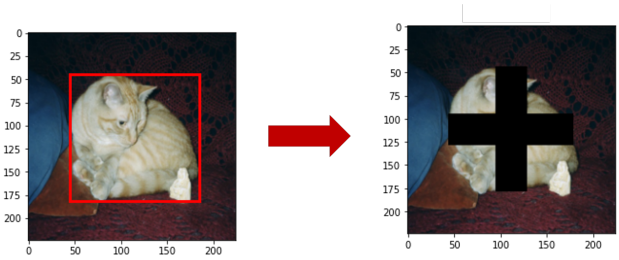
$$R_a(s,s') = \begin{cases} 0 & \Delta_{IoU} \geq 0 \\ -1 - \beta * \sqrt{(\Delta_{IoU})} & \Delta_{IoU} \leq 0 \end{cases} \quad (5)$$

in which β is the coefficient to control the weight of Δ_{IoU}. In the experiment session, we use gird search to search the best $\beta \in [0, 20]$ with the step-size of 4.

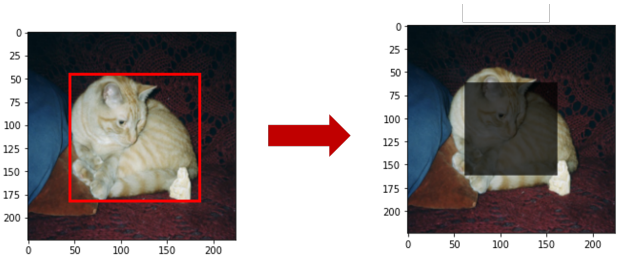
3.4. Inhibition-of-Return Mechanism

Inhibition of return refers to the relative suppression of processing of stimuli that had recently been the focus of attention. It is a very important component of attention, in that it allows a model to rapidly shift the attentional focus over different locations with decreasing saliency, rather than being bound to attend only to the location of maximal saliency at any given time. IOR mechanisms have been widely used in visual attention models to suppress the attended location and avoid endless attractions towards the most salient stimulus(Itti & Koch, 2001).

In the original paper, after an object is localized by the current bounding box and "trigger" action is performed, the agent will marks the region covered by the box with a black cross as shown in Figure 4a. This mark serves as an IoR mechanism by which the currently attended area is prevented from being attended again.



(a) The Process of adding IoR cross.



(b) The Process of adding IoR mask.

Figure 4. Comparison of IoR cross and mask.

However, it causes some problems and has negative influence on the prediction result in the following respect:

- The agent triggered an additional detection in a location where a correct detection was placed before. This happens because the IoR cross leaves some parts of the object visible that may be observed again by the agent.
- Detection missed after the mark added to the image. This happens because the IoR cross covers additional objects in the scene that cannot be recovered.

By analyzing the causes of these issues, we proposed a new IoR Mark and corresponding prediction algorithm in this paper to solve the above two problems. The experiment section will show how they can improve the predicted results greatly. There is a detailed explanation below, and they are summarize in Algorithm 1.

3.4.1. New IoR Mark

We proposed a new IoR mark $mask$ in order to solved the problem caused by the old cross-like IoR mechanism. The $mask$ is created by the following steps:

Step 1:
Given the old bounding box old_bdbox, we can get its edges $[x_{min}, x_{max}, y_{min}, y_{max}]$.

Step 2:
Given the image $image$, we can get its RGB value I.

Step 3:
Create a $mask$, whose shape is the same as I, and set all value to 1.

Step 4:
Set new_area as 0.75×0.75 of the central area of old_bdbox, i.e.
$x'_{min} = x_{min} + 0.25 \times (0.5 \times (x_{min} + x_{max}) - x_{min})$,
$x'_{max} = x_{max} - 0.25 \times (0.5 \times (x_{min} + x_{max}) - x_{min})$,
$y'_{min} = y_{min} + 0.25 \times (0.5 \times (y_{min} + y_{max}) - y_{min})$,
$y'_{max} = y_{max} - 0.25 \times (0.5 \times (y_{min} + y_{max}) - y_{min})$.

Step 5:
Set the new_area's value of $mask$ be 0.2, i.e. the transparency of the new_area in mask is 20%.

Step 6:
Compute $I' = mask \odot I$ to get the new image I', and the $\odot$ is Hadamard product.

The transformation result is shown in Figure 4b.

3.4.2. New Prediction Algorithm

We proposed a new prediction algorithm in order to solved the problem caused by repeated trigger, which is shown in Algorithm 1. Specifically, this algorithm is divided into two parts.

Part 1:
If the given bounding box has never been triggered before, draw a new IoR $mask$ on the image, save the new bounding boxes and move to next iteration.

Part 2:
If the given bounding box has triggered before, enlarge and darker the IoR $mask$ already located in the bounding box, discard the bounding box and move to next iteration. By performing this, we can tell the agent that you have seen this area before, so try to explore some new areas to find new objects rather than focus on the same location.

Here is a example show in Figure 5.

Compared with the original paper that only draw a constant cross, our approach shows a dynamic interaction between the agent and the environment. At the very beginning, the

Algorithm 1 Prediction Algorithm

Input: data $image$, list old_bdbox, list $memory$
Function: $draw\_mask(bdbox)$
Function: $cal\_box(action)$
Function: $select\_action(image)$
repeat
  Initialize $done = false$.
  Initialize $action = select\_action(image)$.
  if $action$ is TRIGGER **then**
    $done = true$
    if $bd\_box \in memory$ **then**
      Do $draw\_mask(old\_bdbox)$
    else
      Update $old\_bdbox \leftarrow new\_bdbox$
      Update $memory \leftarrow new\_bdbox$
      Do $draw\_mask(new\_bdbox)$
    end if
  else
    Compute $new\_bdbox = cal\_box(action)$
    Update $old\_bdbox \leftarrow new\_bdbox$
  end if
  if $step > max\_step$ **then**
    Assign $done = true$
  end if
until $done$ is $true$
Output: list new_bdbox, list $memory$

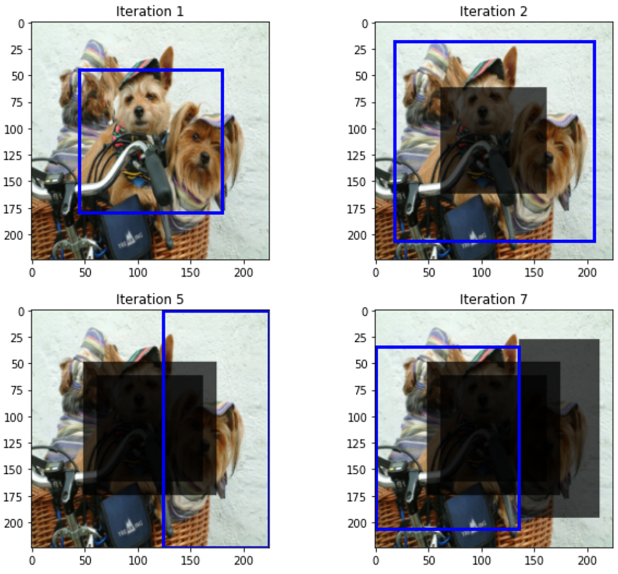


Figure 5. The process of new Prediction Algorithm. In the first picture, the agent triggered the blue bounding box, thus adding a mask shown in the second picture. In the second picture, the agent triggered the same region (If $IoU > 0.5$, two bounding boxes are seemed as the same trigger) as the first one, so the mask is enlarged and darkened. The agent triggered an new area in the third picture, thus adding a new mask as is shown in the fourth picture.

IoR $mask$ is small and thin, which reduces the impact on the agent's ability to see elsewhere in and around the bounding box. If the agent repeatedly trigger one area, the IoR $mask$, or the invisible regions in the bounding box, will gradually becomes larger and thicker, thus preventing the agent from attending only to the location of maximal saliency at any given time.

4. Experiment results

4.1. Experimental Setting

For all the improved parts that we proposed, we conducted extensive experiments on PASCAL Visual Object Classes Detection Dataset and received promising results. Because of the limitation of computational resources and time, we just use VOC2007 Training Set, which contains 2501 images, for training and VOC2007 Validation Set(Everingham et al.), which contains 2510 images as test set. Our train-test ratio is nearly 1:1. We also only tested on first 5 classes, which is [cat, cow, dog, bird, car], instead of whole 20 classes. Because of using fewer training data, our result for baseline (the same agent trained in original paper) may appear worse initially. However, we want to point out that our improved methods are not specifically designed for one dataset or one class. Since their superiority is proved on this small dataset, it is believed that the same improvement will also appear when we apply it to larger dataset and even achieve better performances.

When comparison, we use a reproduction of the agent from original paper as baseline. In each section, we will train several new agents using the methods we proposed, trying to perform better than baseline model. We set IoU threshold at 0.5, which means only predicted bounding boxes that has $IoU > 0.5$ with ground truth will be considered a positive prediction. During all the tests, we use mAP (average precison) and recall as main metrics, which is the same as defined in VOC Competition.

4.2. Reproduction of Original Work

During the test, we first reproduced the results from the original paper. Figure 6 is an example trained on the 'cat' class, the predicted bounding box for each step along with the action taken is shown. It is clear that the agent has learned to take effective actions to transform the bounding box and detect multiple objects by creating cross marks.

We can analyze the detection process of multiple objects by plotting the IoU against number of actions. Figure 7 is an example of an image with three dogs. After crossing an already localized object, the agent is able to localize the other ones. In the IoU plot, we can see that every time the agent initiates a top down search, the IoU is relatively low. As the agent interacts with the image and bounds the object

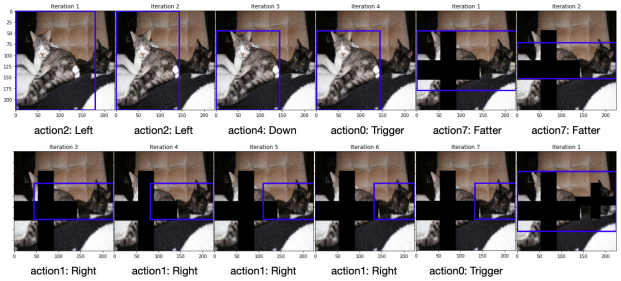


Figure 6. The agent localizing an image with two cat targets

tighter, the IoU rises and reaches a peak for every object detected.

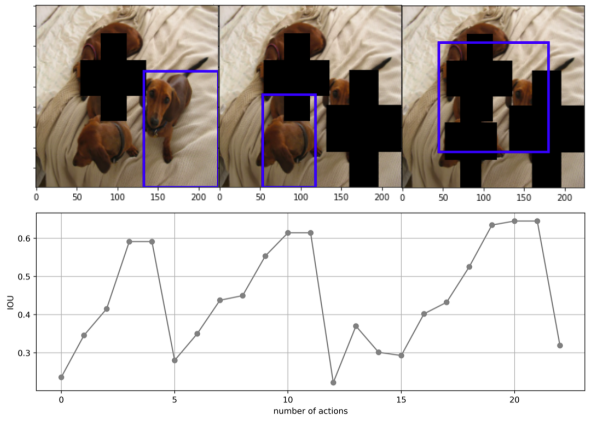


Figure 7. IoU plotted against number of actions taken

4.3. Improved State Space

As stated in experimental setting, after training each network with VOC 2007 training set for 15 epochs and testing in the validation set VOC 2007, we got average precision and recall for 5 objects: cat, cow, dog, bird, and car.

Table 1. Comparison of Average Precision (AP) among different state spaces trained with different feature extractor

	cat	cow	dog	bird	car	mAP
Alexnet	**27.3**	9.1	18.0	**18.0**	18.1	**18.1**
Vgg16	9.1	9.1	**18.2**	9.1	**18.2**	12.7
DenseNet	9.1	9.1	9.1	9.1	9.1	9.1
MobileNet	12.1	9.1	8.8	9.1	9.1	9.6

First observing Table 1 for average precision, we can find that Alexnet ranked first in cat and bird while Vgg16 had the highest average precision in dog and car. The mean average precision of Alexnet is the highest among the 4 models and it is much higher compared with the other 3 models.

Table 2. Comparison of recall among different state spaces trained with different feature extractor

	cat	cow	dog	bird	car	avg
Alexnet	**20.2**	**7.0**	**16.1**	**11.5**	**12.2**	**13.4**
Vgg16	9.6	2.3	13.5	6.9	10.1	8.5
DenseNet	8.6	2.3	9.7	3.6	9.3	6.7
MobileNet	12.6	0.6	9.7	4.6	8.4	7.2

When it comes to Table 2 of Recall, Alexnet won the champion in all objects and it is obvious that Alexnet has the highest average recall. The great performances of Alexnet in both average precision and recall suggest that Alexnet can extract features that are most suitable for agents to learn how to localize objects compared with other models.

These results may be counter intuitive. There might be several underlying reasons why Alexnet and Vgg16 perform better than the other two deeper networks in both average precision and recall. Firstly, the Alexnet with 5 convolutional layers and Vgg16 with 16 convolutional layers are shallower than the other two nets, which produces lower level extracted features. In comparison, the higher level extracted features by deeper networks may be too difficult for agents to observe the details of the image. For example, the receptive field of last layer in DenseNet is 117×117, which is much larger than 17×17 for Alexnet and 27×27 for Vgg16. Therefore, if we use these higher level featurs as our state vector, it may be harder for the agent to observe detail information and analyze its action. Besides, the number of dimensions of feature vector o might also contribute to the result. The feature vector dimensions for AlexNet, Vgg16, MobileNet and DenseNet are 9216, 25088, 28224 and 50176 respectively. If the dimensions of feature vector increases, the possible state spaces will extend exponentially, which makes it harder for agent to learn a good mapping. Therefore, it is also the reason why we can not just extract several lower layers of advanced network to get low level features. In this manner, the feature map will be much larger (e.g. $112 \times 112 \times 32$), which will lead to millions of state space dimensions. It is absolutely harmful. This might also because a large dimension of the feature vector will reduce the effect of history vector h, preventing the agent from learning the knowledge of the past, (h only has 81 dimensions).

In conclusion, the mean average precision and mean recall from the two tables indicate that the use of shallow networks with small output state vector could achieve better result for the agent.

4.4. Expanded Action Space

After extending original 9-action space to 25, we trained two models, keeping all other settings same, and tested it on VOC2007 validation dataset. The mAP result is shown in 3 and recall is shown in 4 by row 1 and 2. We can observe that the performance of our 25-action model is even worse. After some verification tests, we find several possible reasons for this phenomenon. First, as we discussed in reward function section (3.3), agent with original reward function setting sometimes has a problem that keeping transforming its bounding box without trigger. This kind of agent has a behaviour we called 'swipe scores'[2], because it keeps getting +1 even if it only increases IoU a little bit.

Table 3. Comparison of AP among different action spaces

	cat	cow	dog	bird	car	mAP
9-action	18.2	9.1	18.0	9.1	**18.2**	14.5
25-action	9.6	2.3	13.5	6.9	10.1	12.0
9-action + improve	45.2	6.2	35.3	12.6	16.5	23.2
25-action + improve	**49.7**	**18.2**	**36.1**	**13.5**	14.7	**26.5**

Table 4. Comparison of recall among different action spaces

	cat	cow	dog	bird	car	avg
9-action	18.2	1.8	14.6	6.9	10.8	10.5
25-action	15.9	1.5	13.2	5.7	9.5	9.2
9-action + improve	43.4	**14.0**	**35.2**	16.1	**18.5**	25.4
25-action + improve	**50.4**	12.9	31.5	**17.4**	15.7	**25.8**

We observe that this kind of phenomenon becomes more common in 25-action model. It is also easier to understand because we have more α values now. For example, an 25-action agent can keep following three actions from beginning to get infinite rewards: smaller($\alpha = 0.1$), smaller($\alpha = 0.1$), bigger($\alpha = 0.5$). After these three actions, the agent will remain in the same state, and receive reward $r = -1 + 1 \times 0.9 + 1 \times 0.9^2 = 0.71$. This will lead to a loop of all three actions and no localization of object. Also, another problem is that the unbalance distribution of trigger samples. Because in training process, as long as a trigger action is taken, no matter it is from greedy option or non-greedy option, the agent will stop interaction with current image and skip to next one. Therefore, trigger samples actually has an upper bound which is number of all the training images, while samples for other actions do not. So there will be much fewer samples of trigger than other actions. For normal reinforcement learning, it is always not the problem because there are few states that has trigger action which leads to a terminal state. So the state-action pairs $(s, trigger)$ actually consist of a small part of all pairs. However, for this task, it is not the case. Because every crop of image may be the final bounding box, every state has a trigger action. In this situation, the unbalance distribution of trigger samples will be harmful because the agent may not have enough samples to learn the mapping from state to q value of trigger. To solve these two problems, we first switch our reward function proposed in section 3.3 with 0

[2]It means keep a circle of actions to get higher scores

for IoU increasement and -1 for IoU decreasement. Then we use a trick called 'extra trigger', that will force the agent to simulate a trigger action on current state. This extra trigger sample will be added to Memory Replay but will not affect the interaction with agent and environment. The frequency of extra trigger is very important because it will control the percentage of how many trigger samples there will be among all samples. We set the frequency to one exra trigger every 20 samples.

Then, we applied these two improvements on both 9-action model and 25-action model for comparability and get results shown in row 3 and 4 in Table 3 and 4. We can see after apply the improvement, 25 action model keeps showing superiority over 9-action model on the main metric - mAP. It provides the evidence for the effective of both new action space and improvement on extra trigger samples. There are several extra interesting observations. First, after improvement, the superiority of 25-action model over 9-action model is more obvious on mAP than recall. Especially in class cow and dog, the recall of 25-action model is even lower. Although with lower recall, its precision of corresponding class is still higher. It shows a strong evidence that 25-action can predict more precise bounding boxes. Also, there is a speical class - car, where the improvement did not increase its AP. After some analysis, we find that the images in car class contains much more objects than others. For example, there is nearly the same 160 images in class cat and class car. However, car dataset has 800 objects, which is 4 times than 200 objects in cat dataset. Therefore, the bottleneck for predicting car bounding boxes well will be in IoR Mechanism instead of reward or actions. That may be why these improvements did not work for class car.

Here we provided some visualization results about difference predicting process between 9-action model and 25-action on same images in Fig 8. We can see that 25-action model always predict more precise bounding boxes with mostly fewer steps.

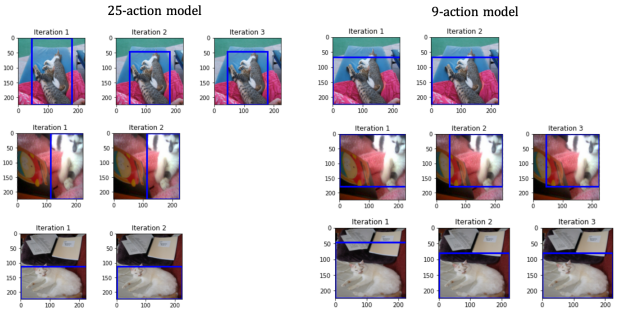


Figure 8. Visualization of predicting sequences of 9 action model and 25 action model.

4.5. Improved Reward Function

In this session, we applied the combination of the sign of variation of IoU and the variation value of IoU to our improved reward function. For all experiments except for the base group, we set $R_a(s, s') = 0$ when $\Delta_{IoU} \geq 0$. We make comparisons between the AP and recall for different categories, the mAP and average recall among all groups based on different β.

Table 5. Average precision with different β.

	cat	cow	dog	bird	car	mAP
Base	18.2	9.1	18.2	9.1	9.1	12.7
$\beta = 0$	36.4	10.7	36.1	16.2	**17.7**	23.4
$\beta = 4$	45.2	**17.3**	35.4	12.0	17.1	**25.4**
$\beta = 8$	44.3	10.7	3.1	11.5	7.0	15.3
$\beta = 12$	**48.6**	1.2	**37.5**	**19.1**	10.7	23.4
$\beta = 16$	18.3	6.4	23.8	4.7	13.9	13.4
$\beta = 20$	28.3	7.7	26.1	15.2	10.0	17.5

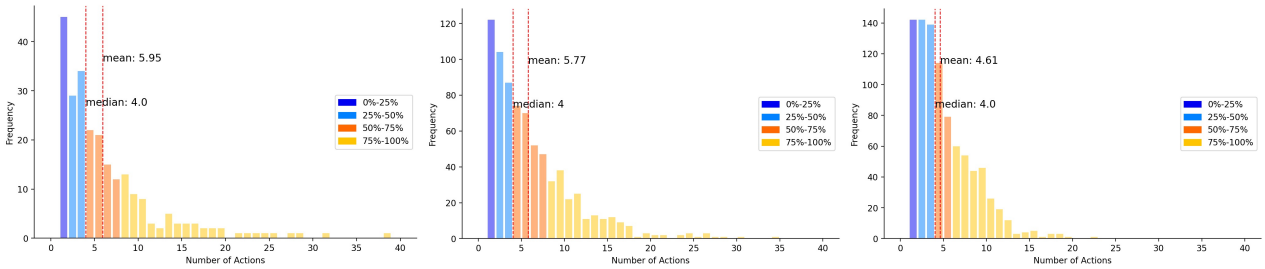
Table 6. Recall with different β.

	cat	cow	dog	bird	car	avg
Base	15.2	2.9	15.4	7.2	9.8	10.1
$\beta = 0$	38.4	13.5	31.5	13.8	21.1	23.7
$\beta = 4$	43.4	16.4	37.4	18.7	22.1	27.6
$\beta = 8$	45.5	24.0	42.3	20.0	23.1	31.0
$\beta = 12$	53.0	24.0	44.6	20.3	**26.3**	33.6
$\beta = 16$	53.5	25.7	40.4	21.0	24.7	33.1
$\beta = 20$	**54.0**	**26.3**	**46.1**	**26.2**	23.8	**36.7**

As is shown in Table 5 and 6, both AP and recall have drastically improvements after changing the reward when $\Delta_{IoU} \geq 0$ from $+1$ to 0. From Table 5, we can see the best APs among different groups are achieved at different β values. The highest mAP is achieved when $\beta = 4$. From Table 6, we can see that the recall of different categories gradually increases as β increases, and the maximum value is achieved when $\beta = 20$.

We also compare the frequency distribution of different number of steps of the trigger events. As can be seen from Fig 9, most trigger events happen when the number of actions is less than 10. At the same time, we discover the baseline reward function has the largest mean and median value of number of steps. After converting $R_a(s, s')$ to 0 when $\Delta_{IoU} \geq 0$, the number of steps before triggering become less in general. It represents a faster localization speed.

We also compare the number of trigger events after improving the reward functions. As can be seen from Fig 10, the number of triggering events has a drastically increase after we improve the reward function. For $\beta = 4$ and $\beta = 20$, which achieve the best mAP and average recall separately,



(a) baseline (b) $\beta = 4$ (c) $\beta = 20$

Figure 9. Comparison of frequency with different β.

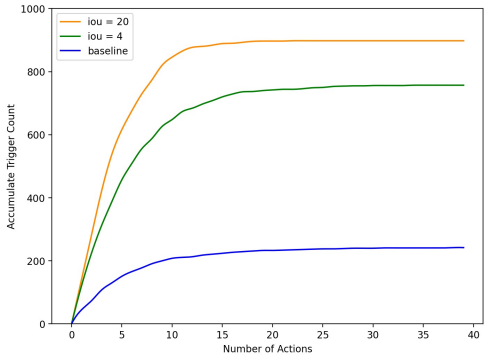


Figure 10. The cumulation of trigger events with different β.

	cat	cow	dog	bird	car	mAP
Cross	42.7	10.2	32.6	17.5	15.5	23.7
Mask	**52.1**	**17.3**	**43.5**	**25.0**	**25.8**	**32.7**
Δ	22.0%	69.6%	33.4%	42.9%	66.5%	38.1%

Table 7. Comparison of mean Average Precision between IoR cross and Mask.

	cat	cow	dog	bird	car	AVG
Cross	47.0	13.5	41.9	16.4	23.1	28.4
Mask	**50.0**	**17.0**	**44.9**	**20.3**	**26.0**	**31.6**
Δ	6.4%	25.9%	7.2%	23.8%	12.6%	11.3%

Table 8. Comparison of mean Recall between IoR cross and Mask.

the reward function when $\beta = 20$ has the most trigger events given the same data-set.

4.6. Inhibition-of-Return Mechanism

In this part, we set the $epoch\_num = 20$ for training, and compare the mean Average Precision and mean Recall between applying the original paper's IoR cross and applying our proposed IoR $mask$ plus Prediction Algorithm in the test set. The result is shown in Table 7 and Table 8.

Generally, it turns out that our proposed IoR mask and new Prediction Algorithm can greatly improve the result of Average Precision and Recall. We can further analyze the experimental results from these two tables.

In Table 7, the mean Average Precision of all classes go up from 23.7 to 32.7, which is improved 38.1%. The increment in Average Precision shows that our method can predict more accurately. This is largely benefited from the new IoR $mask$. By applying the gradually changed IoR mechanism, regions that are triggered at the beginning will not be covered heavily, so the agent can get more information in and around the triggered area to predict the adjacent object more accurately. Therefore, the Average Precision increases, especially in class cow and car, which have relatively more adjacent objects.

In Table 8, the mean Recall of all classes go up from 28.4 to 31.6, which is improved 11.3%. The increment in Recall shows that our method can predict more correct objects. This is benefited from the new Prediction Method. If the agent triggered one region repeatedly, the invisible part of that region will get larger and darker, thus leading the agent to pay more attention to other areas in the image. This algorithm stimulates the agent to explore new regions, so the mean Recall is boosted.

5. Conclusion

In this project, we explored diverse ways to improve an gent on active object localization using deep reinforcement learning and achieved promising results. We first reproduced the training process of original paper, then proposed and implemented improvements in four aspects. In state space, we replaced feature extractor part of Q-Network with several advanced CNN network, then discovered that use of shallow networks with small output state vector, like AlexNet or Vgg16, could achieve better result for the agent. In action space, we proposed a more flexible 25-action model and used extra trigger training to avoid the unbalance of trigger samples, then achieved more efficient and accurate predictions using this extended action space. In reward function,

we put forward 0 positive reward to prevent the agent from wandering around the target object without trigger, then did extensive search to find out the optima value for β is 4 and 20. In IoR mechanism, we replaced the original cross with a mask and put forward new prediction algorithm for multiple objects. The experimental results showed that our new IoR mechanism achieved nearly 40% mAP increasement.

Last but not least, we must point out that our improvement is based on general process of using an agent to do object localization instead of this specific paper. Therefore, it is possible to expand some ideas in this project to other papers and achieve even better results.

References

Ba, J., Mnih, V., and Kavukcuoglu, K. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

Borji, A. and Itti, L. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):185–207, 2012.

Bueno, M. B., Nieto, X. G.-i., Marqués, F., and Torres, J. Hierarchical object detection with deep reinforcement learning. *Deep Learning for Image Processing Applications*, 31(164):3, 2017.

Caicedo, J. C. and Lazebnik, S. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2488–2496, 2015.

Divvala, S. K., Hoiem, D., Hays, J. H., Efros, A. A., and Hebert, M. An empirical study of context in object detection. In *2009 IEEE Conference on computer vision and Pattern Recognition*, pp. 1271–1278. IEEE, 2009.

Endres, I., Shih, K. J., Jiaa, J., and Hoiem, D. Learning collections of part models for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 939–946, 2013.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part based models. tpami, 2010.

Gonzalez-Garcia, A., Vezhnevets, A., and Ferrari, V. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3022–3031, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hosang, J., Benenson, R., and Schiele, B. How good are detection proposals, really? *arXiv preprint arXiv:1406.6962*, 2014.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Itti, L. and Koch, C. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001.

Jie, Z., Liang, X., Feng, J., Jin, X., Lu, W., and Yan, S. Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*, pp. 127–135, 2016.

Juneja, M., Vedaldi, A., Jawahar, C., and Zisserman, A. Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 923–930, 2013.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Lim, J. J., Pirsiavash, H., and Torralba, A. Parsing ikea objects: Fine pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2992–2999, 2013.

Malisiewicz, T., Gupta, A., and Efros, A. A. Ensemble of exemplar-svms for object detection and beyond. In *2011 International conference on computer vision*, pp. 89–96. IEEE, 2011.

Mnih, V., Heess, N., Graves, A., et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pp. 2204–2212, 2014.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Singh, S., Gupta, A., and Efros, A. A. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, pp. 73–86. Springer, 2012.

Torralba, A., Oliva, A., Castelhano, M. S., and Henderson, J. M. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychological review*, 113(4):766, 2006.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057. PMLR, 2015.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

use of shallow networkswith small output state vector could achieve better result forthe agent.

Member Contribution

Table 9. Contribution of Each Member

Name	**Responsible Part**
Jiawei Lu	IoR Mechanism Improvement
Lingyu Zhang	Action Space Improvement
Xinyi Liu	Reward Function Improvement
Yukai Song	State Space Improvement
Zixuan Yan	Action Space Improvement