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** [1] **Yukai Song(ys3493)** [1] **Zixuan Yan (zy2501)** [1]

Abstract

This document provides a basic paper template and submission guidelines. Abstracts must be a single paragraph, ideally between 4–6 sentences long. Gross violations will trigger corrections at the camera-ready phase.

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. 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 on 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 includes conlucison and future improvement.

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, parts of objects, discriminative patches and even salient components of scenes. 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. 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. [18] proposed a branch-and-bound algorithm to find high-scoring regions only evaluating a few locations. Recently, Gonzalez-Garcia et al. [13] proposed an activesearch strategy to accelerate category-specific R-CNN detectors. Another related work is from Divvala et al. [7], 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[3]. 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[33], 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. [35] 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. [22] and Ba et al. [2] 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.

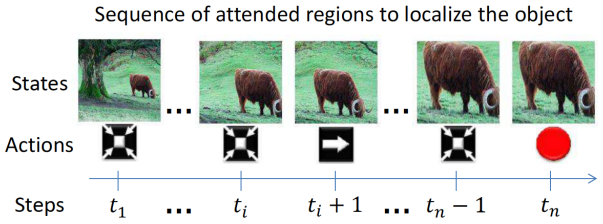


Figure 1. The progress of increasing IoR by iterative analysing current state and taking actions and finally trigger to indicate that an object is found. It is a duplicate figure from the original paper (Caicedo & Lazebnik, 2015).

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.

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

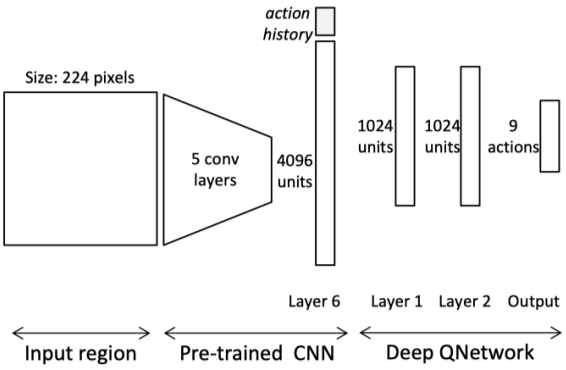


Figure 2. Architecture of the proposed QNetwork. It is a duplicate figure from original paper.

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 $R\left(a, s \rightarrow s'\right)$ is defined for an agent when it takes the action a to move from state s to s' as following:

$$R\left(a, s \rightarrow s'\right) = \operatorname{sign}\left(\operatorname{IoU}\left(b', g\right) - \operatorname{IoU}(b, g)\right) \quad (1)$$

where $\operatorname{IoU}(b, g) = \operatorname{area}(b \cap g) / \operatorname{area}(b \cup g)$ 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 [23] 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.

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, 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.

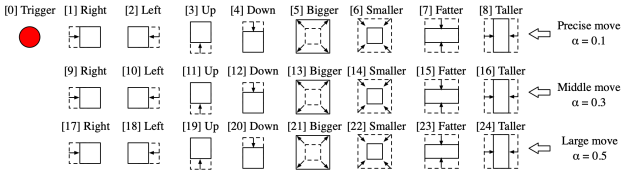


Figure 3. Visualization of 25 actions set. The dotted line represents the bounding box before the action and the solid line represents the bounding box after the action.

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:

$$R_a(s, s') = sign(IoU(b', g) - IoU(b, g)) \tag{2}$$

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

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 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:

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

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:

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

in which α 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

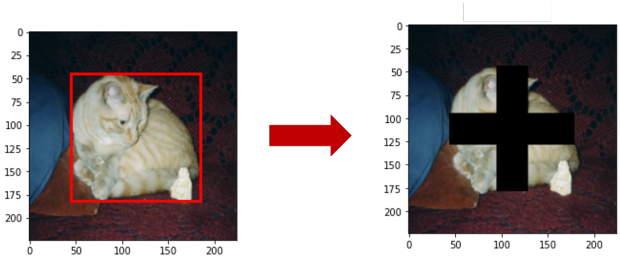
Inhibition of return refers to the relative suppression of processing of stimuli that had recently been the focus of attention. It is a very important component of attention, in that it allows a model to rapidly shift the attentional focus over different locations with decreasing saliency, rather than 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[16].

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.

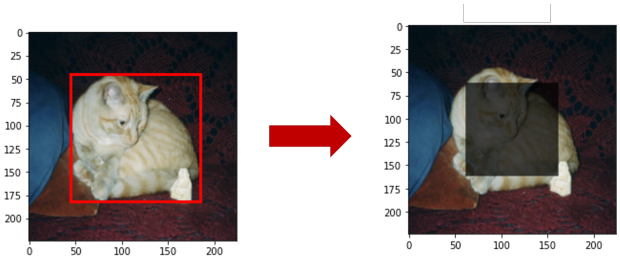
However, it causes some problems and has negative influence on the prediction result in the following respect:

- 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.



(a) The Process of adding IoR cross.



(b) The Process of adding IoR mask.

Figure 4. Comparison of IoR cross and mask.

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:

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 $memory$

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 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.

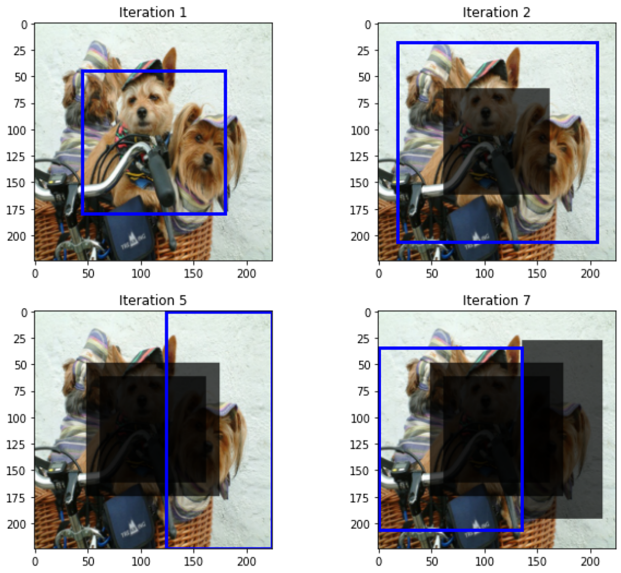


Figure 5. The process of new Prediction Algorithm. In the first picture, the agent triggered the blue bounding box, thus adding a mask shown in the second picture. In the second picture, the agent triggered the same region (If $IoU > 0.5$, two bounding boxes are seemed as the same trigger.) as the first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

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, 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

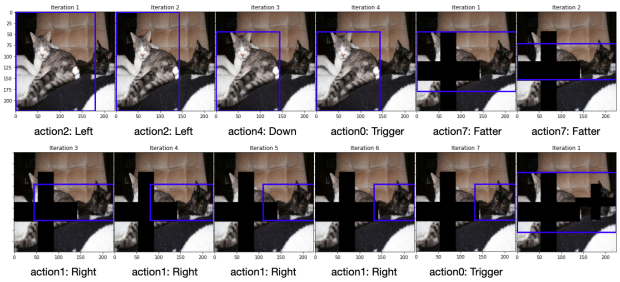


Figure 6. The agent localizing an image with two cat targets

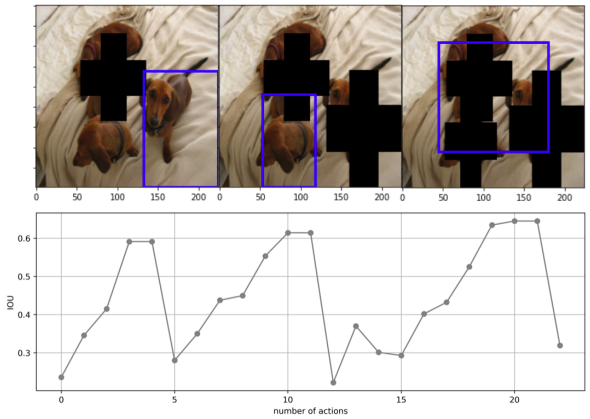


Figure 7. IoU plotted against number of actions taken

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 Paper

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

4.3. Improved State Space

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 among Different State Spaces

	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

Observing the table for comparing 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.

When it comes to the table 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. The low performance of average precision might be because a small number of data in the training set (only used VOC 2007 training set due to the limit of computing resource). Besides, the reward function and action space might have spaces to make a progress.

Table 2. Comparison of Recall among Different State Spaces

	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

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

Table 3. average precision.

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

Table 4. recall.

4.4. Expanded Action Space

4.5. Reward Function Comparison

In this session, we applied the combination of the sign of variation of IoU and

4.6. Inhibition-of-Return Mechanism

In this part, we set the $epoch\_num = 20$, and compare the average precision and recall between applying the original paper's IoR cross and applying our proposed IoR $mask$. The result is shown in Table 5 and Table 6.

It turns out that our proposed IoR mask and new Prediction Algorithm can greatly improve the result of average precision and recall.

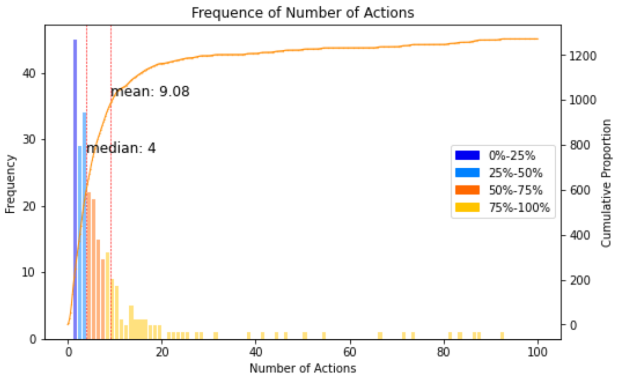


Figure 8.

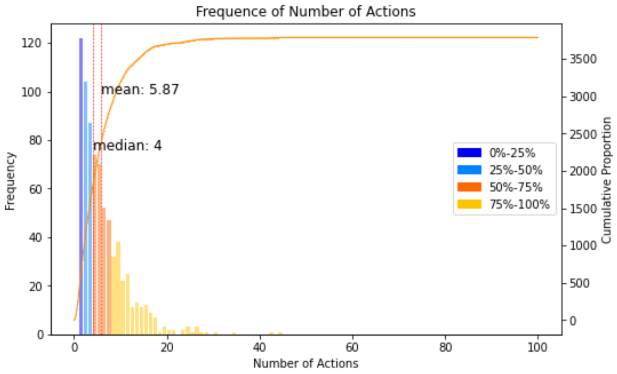


Figure 9.

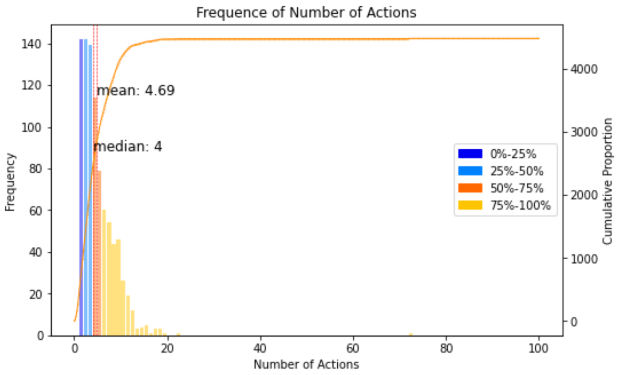


Figure 10.

5. Conclusion

References

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.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE*

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

Table 5. Comparison of Average Precision between IoR cross and Mask.

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

Table 6. Comparison of Recall between IoR cross and Mask.

conference on computer vision and pattern recognition, pp. 770–778, 2016.

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.

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.

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.

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