Image Localization with Convolutional Neural Network and Reinforcement Learning

Chen Huang(ch3371), Chenyu Xi(cx2219), Jiachen Du(jd3488), Tian Xia(tx2178), Yuqin Xu(yx2478)

Abstract

Deep learning techniques have emerged in recent years and led to remarkable breakthroughs in the field of object detection. In this work, we focus on bring up methods for object detection in images. To achieve this, those parts of one image that contain richer information will be focused and enlarged in object detection model. The main idea is to train a reinforcement agent, which has the functional ability to decide the focus area of an image window among candidate regions generated by the image window of last step. Until the reinforcement learning agent reach its final action, the procedure mentioned above will be iterated in terms of hierarchical or dynamic scale analysis strategies. In our work, We adopt convolutional neural network to fetch the image features and build two deep Q learning models. The performance of our methods is tested by a variety of experiments.

1. Introduction

Locating objects from a large number of predefined categories in images is one of the most fundamental and challenging problem in the field of computer vision.[1] In traditional CV models, images are analyzed in place using the method of window scanning. Since it is in different scales, this method of analysis has a weakness that each part of the image do not relate to others, which increase the computational expenses.

When considering the way human look at an image, it is large likely that we focus our eyesight more on the most salient part of the image. Then, in order to obtain full information of the image, we will focus on other parts of the image and try to analyze all the relevant information. Humans instinct behavior mentioned above to gather information from other surroundings introduces the topic of our work. Our method is to use a top-down scanning, which first takes a global view of the whole image and then pay attention to the separated parts of relevant information. With the method of hierarchical object detection introduced, the relevance between regions are more easily addressed. [2]

The models we propose in this project allow a trained reinforcement learning agent to deform a bounding box using simple transformation actions, with the goal of determining the most specific location of target objects.

In the Hierarchical Object Detection Model, the agent recursively divides the image into five sub-regions, including four quadrants and a central region. The agent top-down explores the five sub-regions until it decides that a given region is unlikely to enclose any small objects or it finds an object. In the Bounding Box Transfer Model, we adopt action set containing 9 bounding box transfer movements based on scaling method mentioned in [3], so agent has less chance to go astray even it has made some wrong decisions in past steps during object searching.

Compared to other methods of object detection, our agent is able to extract features from convolutional neural network through the analysis of less number of regions. Also, the representation of higher spatial resolution is even more informative than those proposals by reusing convolutional feature map of the whole image. We trained the model based on the principal of the *Image-Zooms* model [2], which extracts information from each region instead of reuses feature map for different regions of the same image.

The main contribution of our work in the first model is training an agent to use top-down scanning in terms of hierarchical principal, which is elaborated as first takes a global view of the whole image and then pay attention to the separated parts of relevant information. Then we examine the efficiency of hierarchy (extracting features for each region) instead of reusing feature maps for several locations. In the second model, we use scaling method instead of hierarchical actions to generate more precise bounding box. We compare these two thoughts in experiment and conclusion part.

2. Related Work

The current mainstream object detection algorithms are mainly based on deep learning models, which can be divided into two categories: (1) Two-stage detection algorithm. It divides the detection problem into two stages:

firstly generating regional proposals, and then classifying the candidate regions and refining the location. The typical representative of such algorithms is the R-CNN algorithm based on region proposal, such as R-CNN, Fast R-CNN, Faster R-CNN, etc.; (2) One-stage detection algorithm does not require the region proposal stage. It directly generates the class probability and position coordinate values of the object. Typical algorithms are YOLO and SSD.

In the past several years, deep reinforcement learning has driven significant progress in a broad range of problems. Combining with CNN, deep reinforcement learning allows computational models consisting of multiple hierarchical layers to learn fantastically complex, subtle, and abstract representations. R-CNN is a pioneering work based on the region proposal method proposed by R. Girshick et al., 2014 [5]. It first performs regional search using Selective search and then classifies candidate regions using an multi-category SVM classifier. However, when applying the deep learning model to solve object detection problems, it is often necessary to fix the image size, which may cause image loss and affect recognition accuracy. Then, Kaiming He et al. proposed SPP-ne [6] and Spatial Pyramid Pooling Layer to solve this problem. In 2015, mainly draws on the idea of SPP-net, Fast R-CNN [7] is proposed to reduce the time spent by candidate regions in extracting feature vectors using CNN models. The next year, the Faster Region-based Convolutional Network (S. Ren and al. 2016[8]) introduced the RPN (Region Proposal Network) to directly generate candidate regions.

The main performance indicators of the object detection model are detection accuracy and speed. For accuracy, the object detection should consider the positioning accuracy of the object, not just the classification accuracy. In general, the two-stage algorithm has an advantage in accuracy, and the one-stage algorithm has an advantage in speed. However, with the development of research, both types of algorithms have improved in two aspects. Google opened the TensorFlow Object Detection API in 2017 and compared the performance of the mainstream Faster R-CNN, R-FCN and SSD algorithms on the MS COCO dataset. In the paper of Huang et al. 2017 [9], they implemented these three mainstream systems and performed an experimental comparison of some of the main aspects that influence the speed and accuracy of modern object detectors. Recently, Facebook's FAIR opened the Caffe2-based object detection platform Detectron, which implements the latest Mask R-CNN, RetinaNet and other detection algorithms, and gave the Baseline Results [10] of these algorithms.

3. Hierarchical Object Detection Model

Deep Reinforcement Learning (DRL) combines deep learning with reinforcement learning to learn control strategies directly from high-dimensional raw data. Deep Q-Learning Algorithm is one of the algorithms of DRL, which combines Convolutional Neural Network (CNN) and Q-Learning. We aim at achieving a class-agnostic object detection by predicting bounding boxes for certain objects within an image using Deep Q-learning Algorithm.

3.1. VOC 2012

We use PASCAL VOC 2012 for model training, which is a well-known dataset in the field of object detection. This dataset contains approximately 12,000 images with marked object bounding boxes for model training and verification. VOC 2012 is a benchmark dataset for object detection problems, containing 20 categories. We use the 'bird' class for our model training and testing.

3.2. Visual Feature Extraction

We use the *Image-Zooms* model to extract visual features from a image (Figure 1). In this model, each input sub-region image is resized to 224×224, and its visual classification accuracy correspond to the representation depth of feature maps from Pool5 layer of a pre-trained VGG-16 model [11]. As is illustrated in [12], the standard to select feature map is that how large scale of interest the feature map contains. For larger objects, the algorithm will choose deeper feature maps, while for smaller objects, shallower feature maps prove to be more adequate. The Pool5 layer outputs a $7 \times 7 \times 512$ tensor, which is converted to a 25088×1 vector as the region descriptor by the flatten layer.

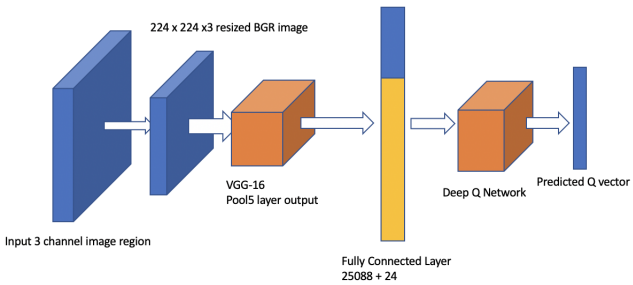

Figure 1. Model Structure

3.3. Deep Q Learning

We present a method for performing hierarchical object detection in image guided by a deep reinforcement learning agent. The most common method to reason a reinforcement learning problem is to represent it as a Markov Decision Process(MDP). The set of states and actions and the rules for the transition from one state to another constitute the MDP. An episode of this process (the movement of the bounding

box in object detection) forms a limited sequence of states, actions, and rewards.

3.3.1. Deep Q Network

The Deep Q Network model takes region descriptor and memory vector as input, containing two fully connected layers of 1024 neurons each [2], with ReLU as their activation functions [13] and are trained with dropout [14]. The 6 neurons fully connected output layer corresponds to the six possible actions of the agent in our model.

3.3.2. State

We store state of each step in a tuple consist of (current feature vector, memory vector). Current feature vector is extracted from the current region image using a pre-trained CNN (VGG-16) [3], and serves as the visual description of the current observed region. The memory vector maps the last four actions have been used. The agent will decide which action to choose based on the concatenation of current feature vector and memory vector.

3.3.3. Action

We achieve bounding box by using mask matrix to indicate which region the bounding box is covering. There are two hierarchical methods to determine the action of zooming bounding box. The first one is the non-overlapping method, as we divided current state into 4 non-overlapping sub-regions plus a center region overlapped with the previous four, and pick one of them as the next region agent will move to.

The five possible movement actions are defined as: left up, right up, left down, right down and central, with regard to the sub-regions we mentioned above. The final action is the terminal trigger action, which indicates the search has finished and the object has been found.

When the first four sub-regions are not overlapped, less steps are required to reach a certain scale of bounding boxes, but the space of possible region is small, and the test results are not precise. We solve this problem by adopting overlapped sub-regions (Figure 2), which performs better finding the object as complete as possible. In this case, we set the size of a sub-region as 3/4 of its ancestor.

3.3.4. Greedy Policy

In model training stage, we adopt ϵ-greedy policy to choose action based on Q value generated by DQN. At the beginning, we choose a larger ϵ to encourage exploration. We decrease the epsilon value by 0.1 at the end of each episode until it reaches 0.1, i.e. we switch from exploration to exploitation as the agent improving its strategy.

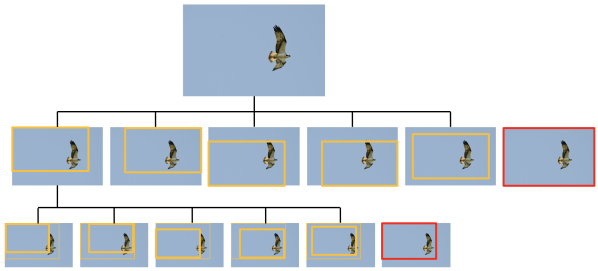

Figure 2. zoom out bounding box with overlapped hierarchical sub-regions

3.3.5. Reward

We use Intersection over Union (IoU) as a metric for our agent's behavior. IoU is calculated by the following equation [2]:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{1}$$

The value of IoU represents area of overlap between the predicted bounding box and the ground-truth bounding box. Thus we choose the reward based on IoU value for current state.

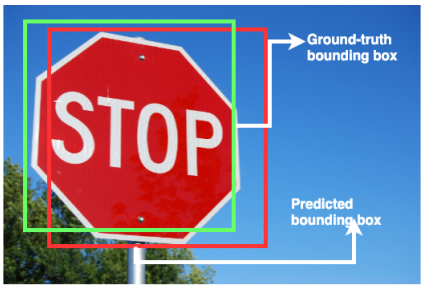

Figure 3. The definition of IoU

Reward represents how good the movement is when the agent move to another sub-region. When the agent moves bounding box and decrease the IoU, it should receive a negative reward, and when the IoU increase after a move, the model offers agent a positive reward. Reward for movement actions is:

$$R_m(s, s') = \mathrm{sgn}(IoU(b', g) - IoU(b, g)) \tag{2}$$

When the agent decides to trigger the terminal state, we compared the IoU between final bounding box and the ground truth box with a IoU threshold value, τ, which is set to 0.5 in our program. If IoU is smaller than the threshold, the agent should receive a high negative reward as it failed to precisely locate the object. On the contrary, the agent receive

high reward when end the search with high IoU. Reward for terminal action:

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

It's is common that one image may contains several objects of same class, in such a case our agent need to learn how to change its target object during searching process. For example, the agent may create a mask has a large IoU over ground truth region of bird 1 while it's looking for bird 2, in such a case we need to calculate reward based on IoU based on bird 1's ground truth region. We achieve this by an array tracking IoU value of all objects of target class with our current region mask, and choose the highest one as our reward metric.

3.3.6. Q-Learning

We want to obtain a function $Q(s, a)$ that predicts best action a in state s in order to maximize a cumulative reward. This function can be estimated using Q value as the label and using Q-learning which updates $Q(s, a)$ using the Bellman Equation:

$$Q(s, a) = r + \gamma max_{a'} Q(s', a') \quad (4)$$

In the equation above, s is the current state and a is the action which will be executed. r is the reward corresponding to the chosen action a. γ is the discount factor, which considers the trade off between immediate reward and future reward. $max_{a'} Q(s', a')$ represents choosing the highest Q value between possible actions from the new state s'.

3.3.7. Experience Replay

One key technology used in DQN is experience replay. Simple Q-learning using neural networks can oscillate or diverge because time-continuous samples are related, not independent. Therefore, DQN uses experience replay, that is, using a memory to store the data that has been experienced, and randomly extracting data from memory for updating to avoid correlation problems.

In our program, we keep a experience pool with the capacity of 500 experiences, and we arbitrarily pick up a mini batch of 100 experiences to sample the target when we update the weights of the Deep Q Network.

3.3.8. Deep Q-Learning Algorithm

Deep Q-learning algorithm is shown below. Our principle is that each region our agent moves to combining with past agent movements can be viewed as a state, and the feature vector of this state is extracted by CNN (VGG16). We can get the action-values of one state through DQN, and then use ϵ-greedy policy to select actions. Then we execute the action and enter a new state. We use IoU to measure the reward and save experience for this step to experience replay pool. Then we updates the DQN weights using stochastic gradient descent to minimize the Q error. After finishing this step, we start at the new state and repeat previous work until the agent triggers the terminal state, or the number of steps overflows.

Algorithm 1 Deep Q-Learning Algorithm

```
Initialize Q, w, ε = 0.9
Create Experience Replay Matrix D
for step ← 1 to number of episodes do
    Initialize S, obtain feature vector φ(S) through Image Zoom Model
    while S' is not terminal do
        Use φ(S) as input, calculate Q value vector using DQN
        Pick action A basing on ε-greedy policy
        Execute A, obtain reward R, new state S and its corresponding feature vector φ(S')
        Store [φ(S),A,R,φ(S')] in D
        Take m samples from matrix D
        if A = 'Trigger' then
            Q_i = R_i
        else
            Q_i = R_i + γ max_{A'} Q(φ(S'), A', w)
        end if
        Loss = (1/m) Σ_{i=1}^{m} Q_i − Q(φ(S), A, w)^2
        Update DQN weights using gradient descend
        ε = ε − 0.1
    end while
end for
```

4. Bounding Box Transfer Model

In the hierarchical detection model above, we use only 5 actions to change the bounding box. However, as we might lose useful information since we dropped the area out of our current region during object searching, those actions may be not precise enough under some circumstances. Therefore, we also build the model proposed by Juan and Svetlana in their paper Active Object Localization with Deep Reinforcement Learning. We named it Bounding Box Transfer Model (9-action Model for short in following paragraphs)

4.1. Approach

The network structure we use in the 9-action model is the same of the hierarchical one. Firstly, we use the Image-Zooms model to extract current region's feature and then use the deep q-learning network to obtain the action. Finally, we use the IoU to evaluate the result of the model. Although

most of the training process is similar with the hierarchical model, the biggest change of the 9-action model our agent is that it will not only choose sub-regions within current region, but also able to choose regions outside using scaling method. Illustrated in Figure 4, the set of actions of the bounding box is composed of eight transformations that can be applied to the box and one action to terminate the search process. Those actions can be divided into four groups: horizontal and vertical moves, scale changing, aspect ratio changing and terminal.

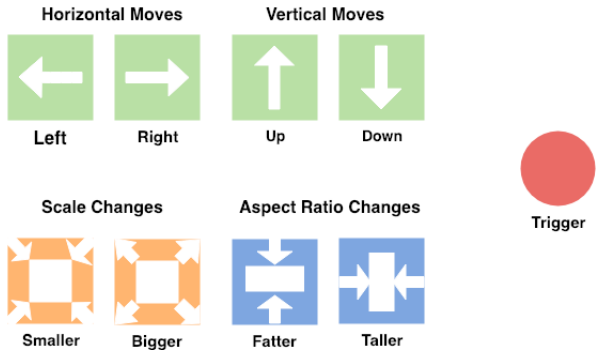

Figure 4. Nine actions of the bounding box

The only action "Trigger" that does not transform the box will terminate the sequence of the current search. Then a new object detecting period for next image will start.

4.2. Comparison

The different actions the agent choose will lead to the different training result. In most of the consequences, the hierarchical model is efficient enough to search the whole image area without any loss. However, when the sub-region mask scale reduction factor is wrongly chosen, there exists chance that agent loses searching area when it moves to smaller regions, which increases the error detection possibility. By switching to the 9-action model, we can avoid such problem. The eight box changing ways can well overlap the whole image area without losing any important part.

However, the 9-action model also has some cons. Too many action choices enlarge the searching time and may add loops during object searching, which may decreases the search efficiency and increase the model training time.

5. Experiment

5.1. Hierarchical Object Detection Model

5.1.1. Training

We trained the object detection model for 'bird' class on VOC 2012 'trainval' dataset. The Deep Q Network is trained by taking 8000 random images as input and running the training process for 30 epoches.

5.1.2. Testing

We test trained model on VOC 2012 test dataset. Although our model is designed to find multiple objects of same class in an image, our testing program only locates the first object with 'bird' label in order to speed up the test process. One object searching episode ends with agent taking 'trigger' action or the step number exceeding our threshold value, which is set to 8 steps to prevent infinite searching.

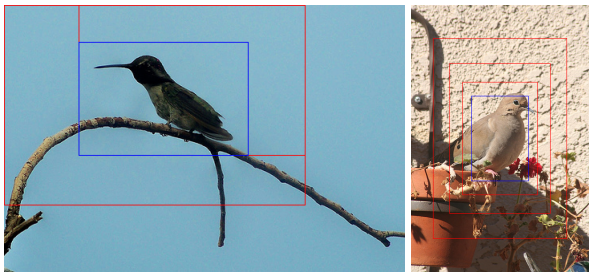

Figure 5. Visualization of search for birds in Hierarchical Model

It turns out that our model works well for most of images in the test dataset. The visualization of successful searches for bird within images is shown in Figure 5. The results indicated that the agent located birds within 5 steps before it triggered terminal state, with output IoU values greater than 0.5.

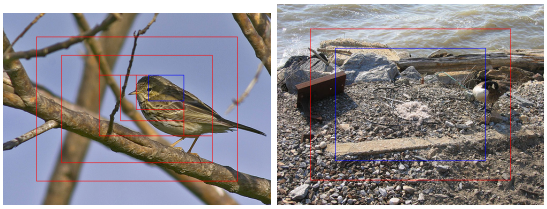

Figure 6. Visualization of failed search in Hierarchical Model

There are also cases when the agent failed to locate right objects. For instance, the agent might fail to trigger the terminal action and continue to crop the region which already has IoU with ground-truth mask larger than terminal threshold value. Meanwhile, the agent might move the bounding box to wrong region or end the searching process prematurely. One possible reason for over-cropping is that the reward setting for terminal state is not proper that the agent is not encouraged to end episode in right state. Another potential risk is the action set we adopt doesn't consider recovery from wrong steps, i.e. as we only create bounding box hierarchically, we cannot find right objects once the agent moves region mask to wrong place, and the following steps are meaningless.

5.1.3. Analysis

In order to get a quantitative sense of the model performance. We define the precise of the trained model with regard to the IoU between the terminal state region and the ground-truth mask. In our work, we test model by feeding in 100 images contains birds and take the average of the IoU outputs. When the model is not trained, the average IoU output is 0.057961152, which is extremely small.

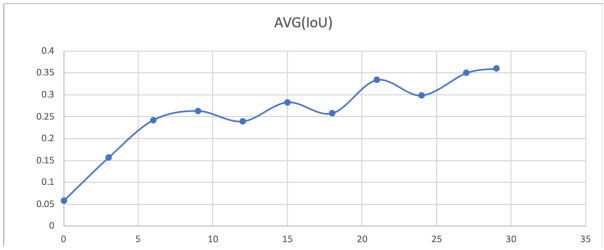

Figure 7. Epoch-IoU Plot

We test the model for each 3 iteration, and plot the average IoU output with regard to epoch number. The output average IoU for the final model is 0.360109787. As seen in the plot, the model performances improves with increasing of total epoch number. However, it also indicates that more epochs is needed to improve our work.

5.2. Bounding Box Transfer Model

5.2.1. Training

We train our 9-action model for images contains aeroplane. There are 328 images that are labeled as 'aeroplane' in our training data. The model is trained for 20 epochs.

5.2.2. Testing

We test trained model on VOC 2012 test dataset. Our testing program only locates the first object with 'aeroplane' label within an image. The object searching ends with agent taking 'trigger' action or the step number exceeding our threshold value, which is set to 10 steps.

5.2.3. Analysis

Our model reaches the mean IoU of 0.413 for localizing aeroplane. The result is shown below.

From the result above, we can see how the actions of the 9-action model is different from the actions of Hierarchical Model. In Hierarchical Model, each action is taken in current region, thus a series of predicting bounding boxes do not overlap. However, in 9-action model, we have actions including moving right and left, zooming in and out, which may make the bounding boxes overlap.

In the right image of Figure 8, the bounding boxes don't

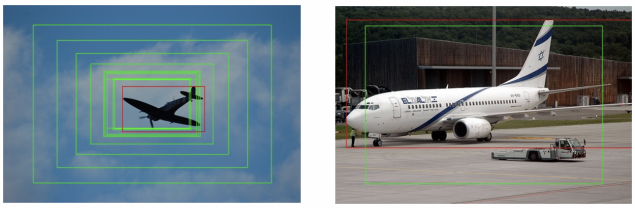

Figure 8. Visualization of search for aeroplane in Bounding Box Transfer Model

Figure 9. IoU of 100 test images

overlap at first, but when the boxes become closer and closer to terminal state, the generated boxes overlap, indicating redundant steps. This may be a reflection that 30 epochs is not enough for our model to learn when to take "trigger" action. Or this may warns us that our reward for terminal state is not set high enough to guide our agent to make terminal action. Thus, our agent hesitates to take "trigger" action or other actions when it approach the terminal state.

Figure 10 displays the IoU of 100 test image where the x axis is the index of the image and the y axis is the IoU of our model prediction on this image. From the figure we can find out that our model fails to find the object in some images where the prediction IoU values are zero. Ignoring the wrong prediction, we can see that the dots equally distributes around IoU=0.45.

During our training for the 9-action model, we notice that it takes a longer time to train the model for each epoch. As we have mentioned above, the 9-action model has 9 different actions including horizontal moves, vertical moves, scale changes, aspect ratio changes and terminal trigger while the hierarchical model only has 6 different actions. Thus, it takes more time for the agent to discover the action-value in the 9-action model, which means that we need to train more epochs to ensure the convergence of action-value and policy.

6. Conclusion

In our work, we apply reinforcement learning to a traditional object detection problem in the field of computer vision.

Generally speaking, we use Convolutional Neural Network to extract the features of images and use Deep Q Network for Q value prediction. Regarding the object localization problem as a decision process, we define different actions and rewards. Thus, the agent is able to obtain its training data through self-playing and use experience replay to fit DQN.

Traditional computer vision methods like YOLO all treat the network as a black box to some extent. It is very hard to explain what the network actually does since we are only able to see network outputs. Our work is meaningful as we opening the black box through interpreting the problem as a combination of feature extraction and action selection, through which the interpretability of network is improved.

There are further efforts needed to be considered to improve our works. Currently, our two models are both not well-trained enough, as training model is a time consuming task. Even though we trained our model using GPU on Google Cloud, it still takes us nearly two days to train a model for 30 epochs. What's more, our current work lacks the comparison of the performance of this two models. To compare their performance, we need to generate several well-trained models for each object detection basing on this two different approaches and then compare their performance on the same object detection.

Appendix

GitHub Link: https://github.com/XiplusChenyu/Object-Location-DQN

Contribution: **Chen Huang** is our team leader, and he is responsible for the coding part of Bounding Box Transfer Model, and **Chenyu Xi** contributes the coding part of Hierarchical Object Detection Model. **Jiachen Du** writes model testing and visualization program, and she analyses the model test results. **Tian Xia** and **Yuqin Xu** have equal contribution to report and they help us setting the google cloud virtual machine.

Acknowledgements

We would like to express our gratitude here to Prof. Chong Li and Prof. Lei Zhang in the Electrical Engineering Department of Columbia University, who provided essential suggestions and guidance for us.

References

[1] L. Liu et al. (2018). *Deep learning for generic object detection: A survey.* [Online]. Available: https://arxiv.org/abs/1809.02165.

[2] Mriam B. Bueno, Xavier Gir-i Nieto, Ferran Marqus, and Jordi Torres. *Hierarchical object detection with deep reinforcement learning.* arXiv:1611.03718v2, 2016.

[3] Juan C Caicedo and Svetlana Lazebnik. *Active object localization with deep reinforcement learning.* In Proceedings of the IEEE International Conference on Computer Vision, pages 24882496, 2015.

[4] Yongxi Lu, Tara Javidi, and Svetlana Lazebnik. *Adaptive object detection using adjacency and zoom prediction.* arXiv preprint arXiv:1512.07711, 2015.

[5] Jonathan Huang, Vivek Rathod, Chen Sun. *Speed/accuracy trade-offs for modern convolutional object detectors.* arXiv:1611.10012v3, 2017.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Region-Based *Convolutional Networks for Accurate Object Detection and Segmentation.* IEEE, 38(1):142-158, 2015

[7] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, A.W.M. Smeulders. *Selective Search for Object Recognition.* IJCV, 2012

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.* arXiv:1406.4729v4, 2014

[9] Ross Girshick, *Fast R-CNN.* IEEE International Conference on Computer Vision, 2016

[10] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* arXiv:1506.01497v3, 2015

[11] E. H. Miller, *A note on reflector arrays (Periodical styleAccepted for publication).* IEEE Trans. Antennas Propagat., to be publised.

[12] J. Wang. *Fundamentals of erbium-doped fiber amplifiers arrays (Periodical styleSubmitted for publication).* IEEE J. Quantum Electron., submitted for publication.

[13] C. J. Kaufman. *Rocky Mountain Research Lab.* Boulder, CO, private communication, 1995.

[14] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa. *Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style).* IEEE Transl, 2, 1987, pp. 740741.