Object Detection with Deep Reinforcement Learning.

Prithvi Raj Singh
School of Computing and Informatics
University of Louisiana at Lafayette
Lafayette, US
prithvi.singh1@louisiana.edu

***Abstract*—The idea of creating a virtual agent that can follow our instructions and learn from it to do a task we want it to do is fascinating, and the same is the idea for using reinforcement learning in object-tracking problems. We formulate the problem as a sequential decision-making process and the agent is equipped with a conventional object detector that detects the object and the agent is trained to make the detection result better. In this paper, we experiment with using the dynamic method for object detection. Our detector, which uses VGG16 pre-trained on ImageNet as a base, is trained using the PASCAL VOC2012 dataset. The agent takes more actions in transforming the bounding box and the reward accumulated. The AP and recall are best for an IoU threshold of 0.5 or lower. We get AP and recall of 36.68 and 59.0 respectively for the 'cat' class at IoU of 0.5. In our experiment the AP and recall declines sharply for IoU threshold for IoU threshold [0.4, 0.5, 0.6, 0.7, 0.8].**

***Index Terms*—Deep Reinforcement Learning (DRL), Average Precision (AP), Recall, IoU, Huber loss.**

I. Introduction

Object tracking is one of the most important tasks in Computer Vision and with the rise of Deep Learning methods like CNN and several variants, the object detection field is getting better and better. The goal of object tracking is to localize the target object inside the frame using discriminating features and track it using spatial-temporal information. There are several advanced Deep Learning (DL) methods like YOLOv7, RCNN, Faster-RCNN, SSD, and many others for precise bounding box detection of the target object. Object tracking is based on sequential object detection and even the state-of-the-art methods will have inaccuracies.

When the object is small it has very limited discriminating factors which makes it harder for many traditional DL methods to properly detect and track the object. Deep Reinforcement Learning (DRL) can be potentially beneficial if it can be designed in the desired way. We can train the agent, which is tasked with localizing the target object, to sequentially detect and track the object with a bounding box. The model that can detect the bounding box for the target object sequentially can track the object online or offline. DRL can be used to localize the target object with geometrical refinement (refinement like transformation to the bounding box for better localization) [2]. The reinforcement learning problem is designed as a process of learning through a feedback loop as in Figure 1. The agent interacts with the environment to achieve a goal.

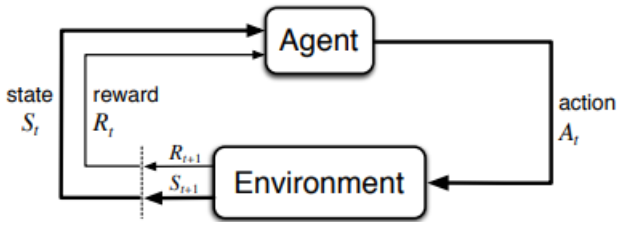

Fig. 1. Agent-environment interaction in RL [7].

Whether it's Multi-object tracking (MOT) or Single object tracking (SOT), precise bounding box detection is important to estimate and forecast the trajectory of the target object. Usual Occlusion and blurriness can degrade the object detection performance and many Deep Learning methods suffer from that. DRL can complement the state-of-the-art object detection models in proper tracking of small, blurry objects through a reward system. Aerial tracking can benefit from DRL-based tracking. For example, we can design a drone-based tracking system that keeps track of the target object. We can reward the drone if the target object localization bounding box has IoU ≥ 0.7, and the negative reward will tell the Drone that it needs to move closer in the direction of the target object.

In this work, we will explore some of the ways Deep Reinforcement Learning has been implemented for object detection, and tracking and try to implement their approach and architecture. The literature review and the architectural implementation of the preexisting work related to DRL for object detection should give us a foundational understanding of DRL applications.

II. Related Works

There has been abundant research in the field of DRL implementation for object detection and tracking. DQN-based approach for object detection was initiated after the success of DeepMind in creating an agent that can play 2600 Atari games. Most of the implementation follows a similar approach and only varies in the ways action dynamics are performed

and how reward is assigned. CNN-based detectors form the base of the detection model and different types of actions are predefined for the agent to take. In [8], Zhou and et.al introduce a DRL-based Multi-Object tracking. In their work, they built a unified prediction-decision network that predicts the bounding box for the target object and a decision network powered with Q-learning takes action to delete or update the position of an object in the frame. This approach helps in making the detection result better.

In [10] Gozena and Ozer introduced a Single object tracking (SOT) DRL method for tracking objects with drones. Similar to the above approach, the SOT tracks a target object by taking a series of actions to locate the object in a tight bounding box. A novel reward function that sets up a policy to reduce the number of steps taken is introduced. Low-resolution images can also be handled using this approach.

In [9] Zhang and et.al introduce DRL-based object tracking using a recurrent convolutional neural network. The RL agent is trained to emphasize the richer part of the frame and learn inter-frame correlation that leads to maximum overall reward. In all of these methods, a policy gradient method is used to update parameters in the direction of the gradient function. The reward signals are assigned in the same way in the above methods - the reward is given based on the closeness between the detected location and the predicted location.

III. Background

A. Markov Decision Process (MDP)

A reinforcement learning (RL) task that satisfies the Markov property is called MDP [book]. In MDP, the reward and the next state depend on the action the agent picks. The agent will take a predefined set of actions and each action will result in a reward and state transition that may or may not be beneficial for the agent to gather maximum reward. To formulate a problem as an MDP problem we define the agent, environment, state, action, and policy.

State Value function in the MDP is expected return from the state following policy π. Policy π is the brain of the agent, and it assigns the highest probability to actions that yield the highest reward $V_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$. *Action-Value function* models an expected return when the agent takes action 'a' and follows policy pi. It is also known as Q-function and is the notion for Q-learning mathematically expressed as $q_\pi(s,a) = \mathbb{E}[G_t|s_t = s, A_t = a]$ we will be using the action-value function for our work. An optimal action-value function $q_\pi(s,a) = \max_\pi q_\pi(s,a)$ models how we can choose policy pi that maximizes the return via picking highest state-action value. The optimal Q-function follows the Bellman optimality equation. $Q^\star(s,a) = \mathbb{E}[r + \gamma \max_{a'} Q^\star(s',a')]$

B. Deep Q Network (DQN)

Originally designed by Deepmind in 2015, DQN was able to solve Atari games by combining RL with deep Q networks

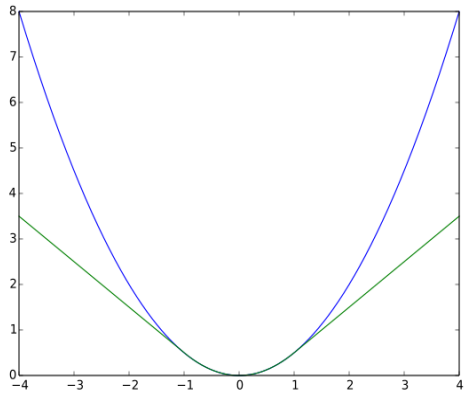

Fig. 2. Huber loss (green), and squared error loss as a function of y-f(x)

[5]. DQN is an enhancement of the Q-learning set-off with a replay memory buffer that stores the transition experience. In DQN a neural network with parameter θ is trained to estimate Q-value. The temporal Difference approach is used to train which updates the value function after each iteration. Update rule follows the equation below, $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_{t+1} + \gamma \left[\max_{a'} Q(s_{t+1},a)\right] - Q(s_t,A_t)$, where α is learning rate and the new Q-value estimation is based on former Q-value estimation added with immediate reward and discounted optimal Q-value of next state. The training update gives us the TD loss represented by $\delta = Q(s,a) - (r + \gamma \max_a Q(s',a))$. TD loss is minimized using the Huber loss function [4].

C. Huber Loss

Huber loss is a combination of Mean Square Error (MSE) and Mean Absolute Error (MAE) and was first introduced by Peter J. Huber [1] in 1964. The goal of Huber loss is to get the potential benefit of both kinds of errors (MSE and MAE). Huber loss acts like MSE when the outliers in the sample are few, and acts like MAE when the outliers are abundant. It is more robust to outliers when the estimates of Q are very noisy [4].

$$L_\delta(y, f(x)) = \begin{cases} 1/2(y - f(x))^2 & |y - f(x)| \leq \delta, \\ \delta|y - f(x)| - 1/2\delta^2 & otherwise \end{cases}$$

If you represent $(y - f(x))$ as 'a' then the Huber function is quadratic for small values of a and linear for large values. For classification purposes, there is a variant of Huber loss called modified Huber shown below that is used. The term $max(0, 1 - yf(x))$ is called hinge loss [6].

$$L(y, f(x)) = \begin{cases} max(0, 1 - yf(x))^2 & yf(x) \geq -1, \\ -4yf(x) & otherwise \end{cases}$$

D. Experience Replay

DQN exploits the idea of experience replay to better approximate the Deep Q Network. It is a memory buffer that

stores the state-action transitions that the agent observes {s, a,r,s'}. A random mini-batch of this state-action transition set is chosen to decorrelate the transition batch since the transition set close to each other will have no significant difference that can help in stabilizing the DQN training procedure.

E. Intersection over Union (IoU)

IoU is the ratio of the intersection area to the union area of the ground truth bounding box and predicted bounding box. IoU greater than 0.5 signifies True Positive.

$$IoU = \frac{|A \bigcap B|}{|A \bigcup B|}$$

F. Average Precision (AP)

Average Precision is a weighted sum of precision at each threshold where the weight is the increase in recall.

$$AP = \sum_{k=0}^{K=n-1} [Recalls(k) - Recalls(k+1)] * Precision(K)$$

G. Policy Gradient Method

In any RL problem, our target is to set up a policy that gives us the maximum expected reward. Policy gradient methods maximize the overall reward by optimizing the policy itself. The policy is parameterized with respect to some constant that affects the policy training. The objective function below depends on this policy. Policy gradient-based methods are good for object detection and tracking problems because tracking is a sequential or continuous process. Policy gradient methods can be on or off-policy-based.

$$J(\theta) = \sum_{s \epsilon S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \epsilon S} d^{\pi}(s) \sum_{a \epsilon A} \pi_{\theta}(a|s) Q^{\pi}(s,a)$$

H. Dynamic Method

Dynamic Method for object detection was introduced by Caicedo and et.al [2]. In this method, an active detection model is used for object localization. The agent focuses its attention on possible regions inside the image frame, as well as the environment, to identify the correct location of the object of interest. The agent performs its action of localizing the target object in a tight bounding box. The reward is given to the agent based on the IoU of ground truth and the detected bounding box of the target object. A series of transformations are applied to the bounding box of a target to get a tight and good overlapping bounding box that yields maximum reward for the agent.

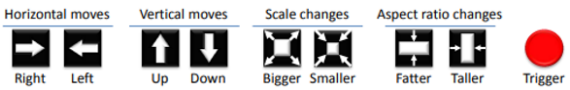

Fig. 3. Transformations applied to BBox [2]

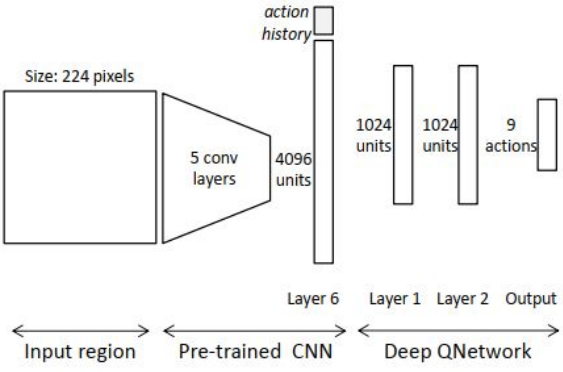

Fig. 4. Dynamic Method Architecture as in [2]

I. Hierarchical Method

The hierarchical Method for object detection was introduced by Bueno and et.al [3]. In this method, the agent is trained to focus on the richer part of the image and zoom in on the target object. There are five predefined non-overlapping regions of interest- four quadrants, and one central region. The policy π is set to allow the agent to look for the target object in the predefined area in as few steps as possible. The basic working philosophy of this method is represented in Figure 4.

IV. Experiment

A. Problem Formulation

Object detection is a sequential decision-making process since the target object in a given video is going to follow a particular trajectory. We can formulate the object detection problem as a Markov Decision Process problem. The MDP problem is parameterized with state S, action A, and reward R values for each episode. An image frame is an environment, and the virtual agent is tasked with the transformation of a bounding box or looking for a target object inside the predefined areas like in the Hierarchical method.

The agent is programmed to follow a particular set of actions and policy π is defined so that it takes optimal action leading to maximal reward. Agent receives reward R_t at a time 't' for taking action. A goal of the agent is to maximize the overall reward which is only possible if the agent finds better detection IoU in fewer transformation steps. The Q-table holds state, and action values and is used during the training process. We will also use the replay buffer to hold all the state-action transition set {s,a,r,s'}.

B. Action and Reward

In general DRL-based object detection, the base detector is a Convolutional Neural Network (CNN), and the RL comes into play for correcting the subpar detection. The RL model is framed in a way that sort of replicates human psychology, positive or negative reinforcement for doing something, as in Fig. 1. For better object detection, the agent in our task has a set of actions to take and each action has some reward with

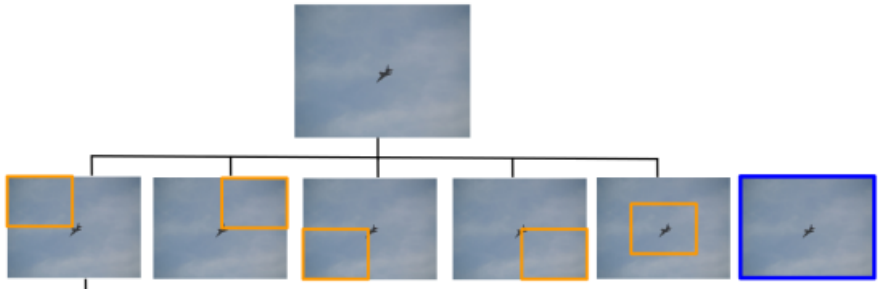

Fig. 5. Hierarchical method in action for object localization [3].

it. Actions are primarily related to transforming the bounding box obtained by CNN-based detection for better detection i.e. IoU ≥ 0.7. If $s_k \epsilon S$ and $a_k \epsilon A$ represent state and action at step k, we can formulate the action in the way shown below,

$$\begin{aligned} s_{k+1} &= f(s_k, a_k, r_k) = (b_{k+1}, d_{k+1}) \\ f_p(b_k, a_k) &= x^k \pm \Delta x^k, y^k \pm \Delta y^k, h^k, w^k \\ &= x^k, y^k, h^k \pm \Delta y^k, w^k \pm \Delta x^k \\ \Delta x^k &= \alpha w^k, \Delta y^k = \alpha h^k \end{aligned}$$

For a function of the state, action, and reward we will have a bounding box associated with a particular frame and transformation dynamics vector d_k. Given a bounding box result from the previous time stamp action 'a' will perform a transformation to the bounding box. There are eight actions that will perform a transformation to the bounding box.

The reward is designated for each action that the agent takes, but for any image, a total reward is calculated based on overall rewards as gathered by performing the bounding box transformation for better IoU with ground truth bounding box. In the training phase, an agent will take several actions to finally figure out the optimal way to get better bounding box detection and overall reward. We define reward as,

$$R_t = \sum_{t-1}^{T} \gamma^t r_t \tag{1}$$

$$R_{s_{(t)}} = \begin{cases} \eta & \text{if } IoU \geq 0.7 \\ \theta & \text{if } 0.5 \leq IoU \leq 0.7 \\ -\eta & \text{otherwise} \end{cases}$$

C. Dynamic Method implementation

Based on the original paper we were able to imitate the experiment for single object detection using the PASCAL VOC 2012 dataset for training and testing purposes. The reward values are different for each 8 different action dynamics. Based on actions taken new IoU is compared with respect to the ground truth and if the IoU is ≥ 0.5 the agent is rewarded with +3 else -3. VGG16 pre-trained on Imagenet is used as a base detection network. The Huber loss function introduced above is used in our Q-model to smooth the training. The Delta δ value is set at 1 for our model's Huber loss. The Q-network is only 3 layers deep. The actions are numbered from 0 to 7 for geometric transformation to the bounding box and action 8 is the trigger that stops the agent from applying any more transformation to the bounding box. The agent takes a maximum of 20 steps to take actions that will yield the maximum total reward for that frame.

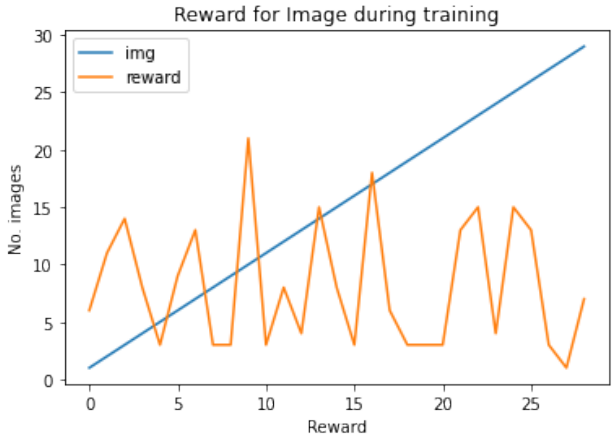

Fig. 6. Reward accumulated for every 30 images.

The reward for 30 images that were tested is based on the actions the agent took to find the target object and tightly localize it. The plot below 7 shows the type of action that was taken the most during the training for all frames used during the training. State-action pairs from 10 previous steps are recorded for the experience replay.

The training process takes more than 22 minutes for each epoch which puts a constraint on the epochs we can train our agent for. Even by using the GPU from Google Colab, the RAM ran out of space to continue the training if we attempted to train for longer epochs. The authors of the dynamic method suggest training for 50 epochs but that is not possible given our computational limit. The IoU we got when testing 100 images is so significant, but we can able to tweak every model to train more accurately and the agent can be modified to better understand the target and localize it.

We performed dynamic method-based detection on the

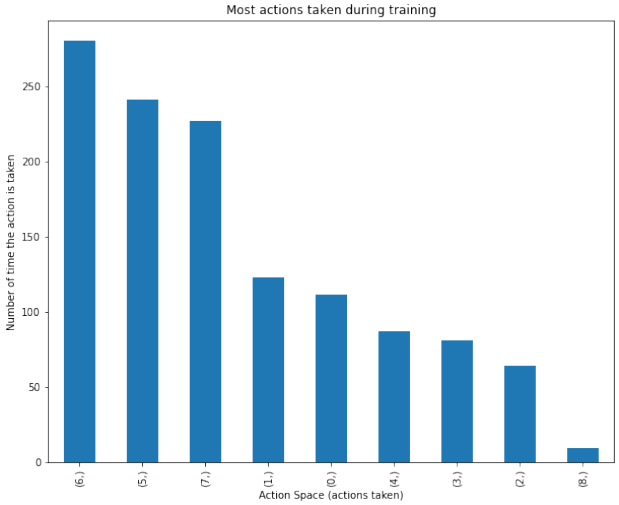

Fig. 7. Actions taken by the agent to collect the reward.

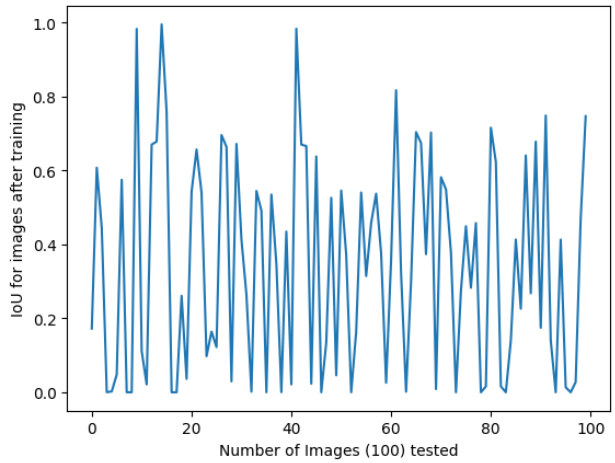

Fig. 8. IoU for 100 images after training for 12 epochs

VOC2012 and VOC2007 datasets and computed the Average precision (AP) and Average Recall (AR) for all 2o classes in the dataset. We set the IOU threshold at [0.4, 0.5, 0.6] and computed the AP and AR after the training. In the table below we only present the AP and AP for the IoU threshold of 0.5 for 5 classes.

Class	cat	motorbike	bus	aeroplane	horse
AP	39.31	32.06	34.43	38.00	32.18
AR	62.18	56.44	58.11	61.26	54.68

For training, we initialize the following parameters - input image size of 224x224, α is 0.2, ϵ is 1.0 with a decay rate of 0.1, the replay memory can hold 1000 state-action transitions, and we train the agent for 15 episodes. We attempted to train the agent for entire classes in the dataset but we encountered that the RAM would run out mid-training and the training would collapse, so due to limited computational means, we settled with training the agent for only 6 classes out of 20

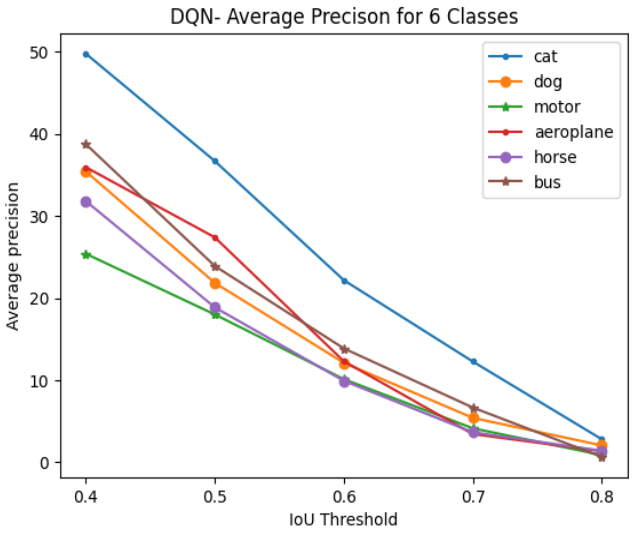

Fig. 9. Precision Curve at different IoU thresholds

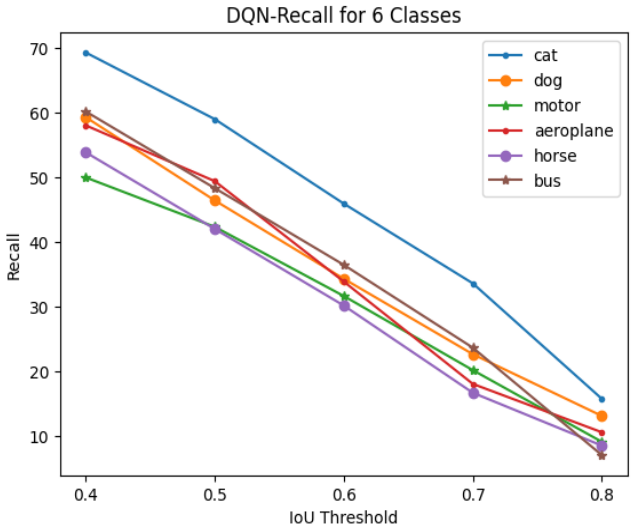

Fig. 10. Recall curve at different IoU thresholds

from the original PASCAL VOC dataset.

Average Precision and Recall show a sharp decline as we increase the IoU threshold which signifies that your DQN agent isn't learning or trained well enough to make better detection and prediction.

D. Dataset

We used the PASCAL VOC 2012 dataset to perform our experiment. The dataset was released in 2012 as a part of the Visual Object Classes Challenge 2012 (VOC 2012). There are 20 classes relating to persons, animals, vehicles, and indoor objects. Around 5717 images from all classes were used for training and 5823 unlabelled images were used for validation. The dataset has a training set of labeled images but

the validation has no label and the trained agent is expected to print the label and detect the object inside the image. We trained our Dynamic method-based model using only the 'airplane' class and in another experiment, we trained the model using all 20 classes from the dataset but performed testing on only six classes. The training and testing require high RAM and a good GPU.

V. Result and Analysis

The average reward for 30 images we tested from the first experiment, presented in Fig 6, is around 8 and from bar chart Fig 7 we can see that the number of trigger action, which stops the agent from transforming the bounding box (BBox), is least among all actions taken which implies that the agent had to quit bounding box transformation before being able to find the best overlapping window for the target object. Probably increasing the max steps will allow the agent to do some more bounding box transformation and find better IoU and hence more reward. In the implementation, the reward is either +1 or -1 but we can make better policy π for the agent if we assign $3^r d$ reward of 0 if the agent finds an overlap of 0.5 or greater, in this way, the agent might follow better policy. The agent constantly keeps changing the policy at each state to get more reward so we should allow the agent to take more optimal action if it finds IoU of $\geq$ 0.5. The trigger actions should also stop the agent from more BBox transformation if at any state the agent gets an overlap of 70 percent or more. This will allow the agent to keep maximum reward and be conservative.

In our other experiment, the DQN is still only 3 layers and we are using the VGG16 base as a feature extractor. The training for each class's single episode takes around 2 mins on Google Colab so the total experiment took around 4 hours to finish and get the visualization result. There was a sharp decline in the IoU for all the object classes as we increased the IoU threshold from 0.4 to 0.8 with an increment of 0.1. This tells us that we need better policy and train the agent for a higher number of episodes with possibly a larger training dataset. We believe that we should put some constraint on the agent's free movement of the bounding box to localize the object better and faster.

The implementation of the dynamic method was challenging and time-consuming since there were more actions to be taken. We believe that the use of DRL for object detection problems is in itself tricky and we might not be benefiting from the potential of reinforcement learning instead we are interfering with the detection power of deep neural networks. Even though the results aren't great, the implementation showed us how we can apply DRL to problems that are not trivial to the reinforcement learning area. It might be beneficial to experiment using other pre-trained models like ResNet, Inceptionv3, and EfficientNet as feature extractors. Training the agent for more episodes and a larger dataset will in itself not be beneficial if we don't formulate a better policy. The reward function along with the policy of the agent needs to be redefined so we can get better results.

VI. Future Work

Working on this project has been a really rewarding experience as we were exposed to the implementation of reinforcement learning for object-tracking problems. We would like to continue working on self-design and implementation of DRL for object tracking problems. It will be impactful if we can create or use some virtual simulated environment where we can see the visualize the agent in action. In the physical world we can program a drone to follow an object and use reinforcement learning techniques to enable the robot to take action i.e. follow the target object if the bounding box obtained by the agent gets smaller than the bounding box obtained from the state-of-the-art object tracking method. Our very immediate work will be to optimize and redefine the policy and reward function in the dynamic method so we can achieve better precision-recall for our dataset. We will also emphasize using our own dataset that contains a tiny ball as a target object to see the performance of DQN for detecting smaller objects while not using any pre-trained model as a base network.

One other work that we want to do is to explore the use of reinforcement learning in trajectory prediction tasks. If applicable, reinforcement learning can benefit the LSTM or CNN-based trajectory prediction. The agent will be trained to correct or interpolate the missing links in the trajectory.

VII. Conclusion

In this paper, we experiment using the dynamic method for object detection using the PASCAL VOC2012 as a train and validation dataset. The dynamic method applies a transformation to the bounding box obtained from the base detection model hence at each step it tries to maximize the reward by taking action but it also shows bad performance in localizing the target object. The hierarchical method that we weren't able to implement has predefined regions where the agent tries to find the object and localize, but the dynamic method allows for free movement of the bounding box which subsequently takes a longer time to train and still performs worse. The average precision (AP) and the Recall for VOC2012 object classes like cat, dog, motorbike, and airplane degrade as we increase the IoU threshold. We can conclude that more work needs to be done to optimize the object detection.

Acknowledgment

Credit to all the online articles that help in understanding the problem. We thank Dr. Maida for all his effort in helping us understand the fundamentals of reinforcement learning and how object detection problems can be formulated as RL problems. We also want to thank all of our colleagues in the class who listened and offered their ideas for solving our problem.

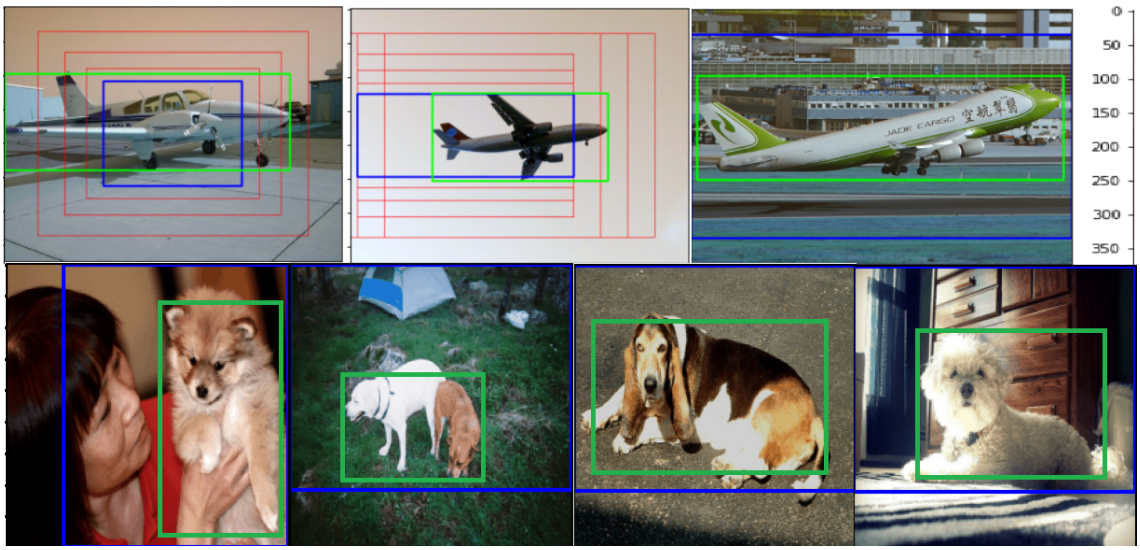

Fig. 11. Results from the experiment that shows the agent trying bounding box transformation to tightly localize the target object. The green bounding box is ground truth, the blue bounding box is the detected one and the red bounding box shows the transformation applied to the predicted bounding box.

REFERENCES

[1] Huber, Peter J. "Robust estimation of a location parameter." Breakthroughs in statistics. Springer, New York, NY, 1992. 492-518.

[2] Caicedo, Juan C., and Svetlana Lazebnik. "Active object localization with deep reinforcement learning." Proceedings of the IEEE international conference on computer vision. 2015.

[3] Bueno, Míriam Bellver, et al. "Hierarchical object detection with deep reinforcement learning." Deep Learning for Image Processing Applications 31.164 (2017).

[4] Paszke, Adam. "Reinforcement Learning (DQN) Tutorial." Reinforcement Learning (DQN) Tutorial, https://pytorch.org/tutorials/intermediate/reinforcementqlearning.html.

[5] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[6] "Huber Loss." Wikipedia, Wikimedia Foundation, 6 Oct. 2022, https://en.wikipedia.org/wiki/Huber loss.

[7] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[8] Ren, Liangliang, et al. "Collaborative deep reinforcement learning for multi-object tracking." Proceedings of the European conference on computer vision (ECCV). 2018.

[9] Zhang, Da, et al. "Deep reinforcement learning for visual object tracking in videos." arXiv preprint arXiv:1701.08936 (2017).

[10] Ozer, Sedat. "Visual object tracking in drone images with deep reinforcement learning." 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

[11] Samiei, Manoosh, and Ruofeng Li. "Object Detection with Deep Reinforcement Learning." arXiv preprint arXiv:2208.04511 (2022).