Reinforcement Learning for Improving Object Detection

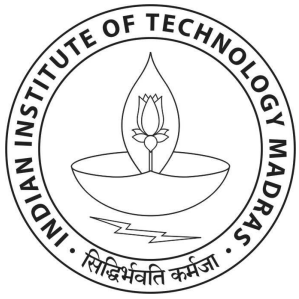
A Project Report

submitted by

SIDDHARTH NAGAR NAYAK

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Reinforcement Learning for Improving Object Detection**, submitted by **Siddharth Nagar Nayak**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Balaraman Ravindran
Research Guide
Professor
Dept. of CSE
IIT-Madras, 600 036
Place: Chennai
Date: 30th May 2020

ABSTRACT

KEYWORDS: Reinforcement Learning, Object Detection, Camera Parameters

The performance of a trained object detection neural network depends a lot on the image quality. Generally, images are pre-processed before feeding them into the neural network and domain knowledge about the image dataset is used to choose the pre-processing techniques. In this thesis, an algorithm called ObjectRL is introduced which chooses the amount of a particular pre-processing to be applied to improve the object detection performances of a pre-trained network. The main motivation for ObjectRL is that an image which looks good to a human eye may not necessarily be the optimal one for a pre-trained object detector to detect objects. We also introduce an algorithm called hImgRL which is trained with reinforcement learning with a human-based reward system to recover back a human-pleasing image.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

8figure.caption.9

ABBREVIATIONS

RL	Reinforcement Learning
MDP	Markov Decision Process
PPO	Proximal Policy Optimization
SSD	Single Shot Detector (Object Detection Algorithm)
YOLO	You Only Look Once (Object Detection Algorithm)
IoU	Intersection over Union
ReLU	Rectified Linear Units
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ConvNet	Convolutional Neural Network

NOTATION

S	Set of states
A	Set of actions
R	Reward function
γ	Discount factor
π	Policy
π^*	Optimal Policy
V	Value Function
$I(x, y)$	Pixel intensity at co-ordinates (x,y) for a particular channel

CHAPTER 1

INTRODUCTION

1.1 Overview

With the advent of convolutional neural networks, object detection in images has improved significantly giving rise to several object detection algorithms like YOLO [Redmon *et al.* (2015)], SSD [Liu *et al.* (2015)], etc. Most object detection networks work with raw image pixels as inputs. The networks are highly nonlinear in nature and thus the output predictions depend a lot on the image parameters like brightness, contrast, etc. [Osadchy and Keren (2001); Linderoth *et al.* (2013); Osadchy and Keren (2004); Maier *et al.* (2011)]. In real-world scenarios, camera parameters like the shutter-speeds, gains, etc. with which the images are taken, matter a lot in the performance of an object detection network [Andreopoulos and Tsotsos (2012)]. A photographer changes a lot of parameters like the shutter speed, voltage gains, etc. while capturing images according to the lighting conditions and the movements of the subject. In autonomous navigation, robotics, etc. there are several instances where the lighting conditions and the subject speed changes. In these cases, using fixed shutter speed and voltage-gain values would result in an image which would not be conducive for object detection. Most cameras rely on the built-in auto-exposure algorithms to set the exposure parameters of the camera. Although the images obtained from these auto-exposure algorithms may be *pleasing* to a human eye, they may not be the best image to perform object detection on. Also, most of the object detection networks are trained using images from a dataset which are captured either by using a single operation mode or no control over the parameters of the camera. Thus, a pre-trained network may have a larger affinity towards images captured with similar parameters as the ones in the dataset it was trained on.

1.2 Contributions

In this thesis, to tackle the problem of sudden variations in the photography conditions, we propose to train a Reinforcement Learning (RL) agent to digitally transform images in real-time such that the object detection performance is maximised. Although we perform experiments with digital transformations, this method can ideally be extended to choose the camera parameters to capture the images by using the image formation model proposed by Hasinoff *et al.* (2010). We train the model with images which are digitally distorted, for example: changing brightness, contrast, color, etc. It should be noted that we do not necessarily want the agent to recover the original image.

CHAPTER 2

BACKGROUND

In this chapter we briefly review the literature and the existing methods related to image modifications for object detection improvement provide the necessary background.

2.1 Related Work

Bychkovsky *et al.* (2011) present a dataset of input and retouched image pairs called MIT-Adobe FiveK, which was created by professional experts. They use this dataset to train a supervised model for color and tone adjustment in images. The main motive of this work is not inclined towards improving object detection but is more focused towards training a model to edit an image according to the user preferences.

Wu and Tsotsos (2017) create a dataset of images taken with different combinations of shutter speeds and voltage gains of a camera. They create a performance table which is a matrix of mean average precision (mAP) for detection of objects in images taken with different combinations of shutter speed and gains. To choose the optimal parameters to capture images, they propose to choose the combination which gives the maximum precision. One of the problems with this method is that a dataset with images taken with different combinations of shutter speeds, voltage gains and illuminations has to be manually annotated with bounding boxes around the objects which is quite time-consuming. Also, the dataset consists of images with static objects. Thus, the effect of changing shutter speed is just on the overall brightness of the image. But one of the main reasons for changing shutter speed while capturing images is to increase (for artistic purposes) or (preferably) decrease motion blur in the moving objects.

Park *et al.* (2018) propose a reinforcement learning based method to recover digitally distorted images. The authors model the agent to take actions sequentially by choosing the type of modification (brightness, contrast, color saturation, etc.). The main motive of this model is to recover back the distorted images. The reward for the agent is

the difference of mean square difference of the images at the current time step and the previous time step. This is slightly similar to our *hImgRL* model in the sense that both the models try to recover distorted images but our model is trained with a human-based reward system which we describe in section 3.1.2. Also, this work is quite different from our *ObjectRL* model as our main motive is to maximise the performance of a pre-trained detector.

Reinforcement Learning has been used in conjunction with computational photography in recent works by Yang *et al.* (2018) and Hu *et al.* (2018) where the authors train RL agents to either capture images or post-process images in such a way that the resultant image is *visually pleasing*. The agent gets a reward from the users according to their preferences of exposures on cameras in the former one whereas in the later one the agent receives a reward based on the discriminator loss of a Generative Adversarial Network [Goodfellow *et al.* (2014)].

Another area of research orthogonal to ours is using reinforcement learning to obtain region proposals for object-detection and object-localization [Mathe and Sminchisescu (2014); Koenig *et al.* (2018); Caicedo and Lazebnik (2015); Mathe *et al.* (2016)]. The main motivation is to make the agent focus its attention toward candidate regions to detect objects by sequentially shifting the proposed region and rewarding the agent according to the Intersection over Union ($IoU-$explained in Section 2.3.4).

2.2 Reinforcement Learning

Reinforcement learning (RL) tries to solve the sequential decision problems by learning from trial and error. Considering the standard RL setting where an agent interacts with an environment $\mathcal{E}$ over discrete time steps. In the time step t, the agent receives a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to its policy π, where $\mathcal{S}$ and $\mathcal{A}$ denote the sets of all possible states and actions respectively. After the action, the agent observes a scalar reward r_t and receives the next state s_{t+1}. The goal of the agent is to choose actions to maximize the cumulative sum of rewards over time. In other words, the action selection implicitly considers the future rewards. The discounted return is defined as $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$, where $\gamma \in [0, 1]$ is a discount factor that trades-off the importance of recent and future rewards.

RL algorithms can be divided into two main sub-classes: Value-based and Policy-based methods. In value-based methods, values are assigned to states by calculating an expected cumulative score of the current state. Thus, the states which get more rewards, get higher values. In policy-based methods, the goal is to learn a map from the states to actions, which can be stochastic as well as deterministic. A class of algorithms called actor-critic methods [Konda and Tsitsiklis (2000)] lie in the intersection of value-based methods and policy-based methods, where the critic learns a value function and the actor updates the policy in a direction suggested by the critic.

2.2.1 Value-based Method

In value-based methods for RL, we have a value function $V(s)$ which intuitively measures how good is it to be in a particular state. By definition, value function is the expected discounted rewards that the agent will get by following a specific policy.

$$V^{\pi}(s_t) = \sum_{\tau=t}^{T} \mathbb{E}_{\pi_\theta}[\gamma^{\tau-t} r(s_\tau, a_\tau)|s_t] \tag{2.1}$$

The objective is to find a policy π^* which maximizes the expected rewards.

$$V^*(s) = \max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{H} \gamma^t r(s_t, a_t, s_{t+1})|\pi, s_0 = s\Big] \tag{2.2}$$

2.2.2 Policy-based Method

In policy-based methods for RL, instead of learning a value function over the states, a policy is learned directly which maps states to actions. The policy learned can be represented as:

$$\pi(a|s, \theta) = P(a_t = a|s_t = s, \theta_t = \theta) \tag{2.3}$$

which means that the policy π is the probability of taking action a when at state s and the parameters of function approximator are θ.

REINFORCE

REINFORCE [Williams (1992)] is a policy-based algorithm which iteratively updates the agent's parameters by computing the policy gradient.

$$\Delta\theta_t = \alpha\nabla_\theta \log \pi_\theta(s_t, a_t)R_t \tag{2.4}$$

Where α is the learning rate, R_t is an estimate of $Q^\pi(s_t, a_t)$. We can also use baselines to reduce the variance of this estimate by keeping it unbiased by subtracting a learned function of the state $b_t(s_t)$. Therefore the resulting gradient is $\nabla_\theta log\pi(a_t|s_t;\theta)(R_t - b_t(s_t))$. A learned estimate of the value function which is commonly used is $b_t(s_t) \approx V^\pi(s_t)$.

Proximal Policy Optimization

We use PPO Schulman *et al.* (2017) which is a type of actor-critic method for optimising the RL agent. One of the key points in PPO is that it ensures that a new update of the current policy does not change it too much from the previous policy. This leads to less variance in training at the cost of some bias, but ensures smoother training and also makes sure the agent does not go down an unrecoverable path of taking unreasonable actions. PPO uses a clipped surrogate objective function which is a first order trust region approximation. The purpose of the clipped surrogate objective is to stabilize training via constraining the the policy changes at each step.

2.3 Object Detection

Object recognition is an essential research direction in computer vision. Most of the successful object recognition algorithms use deep convolutional neural networks which are trained to give the co-ordinates of the bounding boxes around the objects. With the advent of faster processing units training larger neural networks with more complicated network structures has become possible to obtain state-of-the-art models even surpassing humans in tasks like image classification.[He *et al.* (2015*a*),He *et al.* (2015*b*)].

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (ConvNet) are a type a network architecture inspired by the connectivity patterns of neurons in human brain in the visual cortex. Individual neurons respond only to stimuli in a restricted region of the visual field nown as the receptive field. The collection of all such receptive fields overlap to cover the entire visual area.

The advantage of ConvNets over feed forward networks is that a ConvNet is able to capture the spatial dependencies of an image through the application of filters. Also, ConvNets have much lesser parameters to learn as compared to a feed forward network of the same size. Object detection networks take an image as an input and output the co-ordinates of the bounding boxes of the detected objects, the prediction of class of the object and the confidence of the prediction.

2.3.2 Single Shot Detector

In Single Shot Detectors (SSD), as the name suggests, the image needs to be passed only once to detect multiple objects in an image as compared to the two passes in regional proposal network (RPN) based networks. The first pass is for generating region proposals and the second pass is for detecting the objects in region proposals. Thus, SSD is much faster than RPN-based approaches.

In SSD, bounding box predictions from the last few layers of the network where each is responsible for progressively smaller bounding boxes. The final prediction is the union of all the predictions made by these layers.

2.3.3 You Only Look Once

You Only Look Once (YOLO) is an object detection system targeted for real-time processing. The predictions for bounding boxes and object classed are made from a single network which can be trained end-to-end. Like SSD, YOLO is also a single pass detector as it predicts the bounding boxes and the class probabilities for each image with a single network pass.

2.3.4 Intersection over Union

Object detection networks output co-ordinates of bounding boxes. To decide whether an object is detected or not, the Intersection over Union (IoU) criteria is used. Intersection over Union is the ratio of area of overlap and area of union of the predicted and the ground truth bounding boxes. Let p be the predicted box, and g be the ground truth box for the target object. Then, IoU between p and g is defined as $IoU(p, g) = area(p \cap g)/area(p \cup g)$. Generally, if $IoU > 0.5$ an object is said to be a True-Positive.

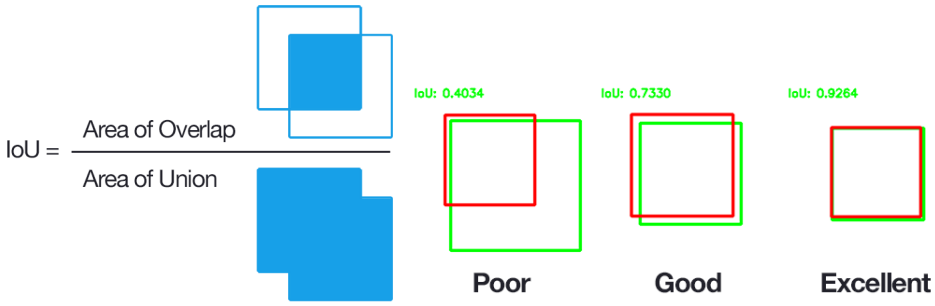


Figure 2.1: Intersection over Union[1]

2.3.5 Image Distortions

Different parameters of an image like brightness, contrast and color can be changed digitally. We describe the formula used to transform the pixel intensity (I) values at the co-ordinates (x, y). We assume distortion factor $\alpha \geq 0$

Brightness

The brightness of an image can be changed by a factor α as follows:

$$I(x, y) \leftarrow \min(\alpha I(x, y), 255) \tag{2.5}$$

Figure 2.2: Variation in images with varying brightness distortion factor α from 0 to 2 in steps of 0.1.

Color

The color of an image is changed by a factor α as follows:

We evaluate the gray-scale image as:

$gray = (I(r) + I(g) + I(b))/3$, where I(r), I(g) and I(b) are the R, G & B pixel values respectively.

$$I(x, y) \leftarrow min(\alpha I(x, y) + (1 - \alpha)gray(x, y), 255) \tag{2.6}$$

Figure 2.3: Variation in images with varying color distortion factor α from 0 to 2 in steps of 0.1.

Contrast

The contrast in an image is changed by a factor α as follows:

$$\mu_{gray} = mean(gray)$$

$$I(x, y) \leftarrow min(\alpha I(x, y) + (1 - \alpha)\mu_{gray}, 255) \tag{2.7}$$

Figure 2.4: Variation in images with varying contrast distortion factor α from 0 to 2 in steps of 0.1.

CHAPTER 3

ALGORITHMS

3.1 hImgRL

3.1.1 Problem Formulation

We cast the problem of image parameter modifications as a Markov Decision Process (MDP) [Puterman (1994)] since this setting provides a formal framework to model an agent that makes a sequence of decisions. Our formulation considers a single image as the state. To simulate the effect of *bad* images as well as increase the variance in the images in a dataset, we digitally distort the images. These digital distortions are carried out by randomly choosing α for brightness distortion. We have a pre-trained object detection network which could be trained either on the same dataset or any other dataset. The main idea of the *hImgRL* is to suggest a particular change in brightness to the input image such that a human can identify the objects present in the image.

3.1.2 MDP for hImgRL

The Markov Decision Process Framework for the hImgRL setting is defined in this section.

States: The states for the agent are $64 \times 64 \times 3$ images from the CIFAR-10 dataset Krizhevsky (2009) which are distorted by changing the brightness by random factors $\alpha \in [0, 2]$, as described in Section 2.3.5 during training.

Actions: The agent can choose to change the brightness by choosing one of the 41 values from $\mathcal{A} = \{0, 0.05, 0.1, \cdots, 1.90, 1.95, 2\}$. Here, $a_t = 1$ does not change the image, $a_t = 0$ gives a dark image and $a_t = 2$ gives an image which is quite bright. The variation of the distorted images is shown in the Figure 2.2.

Reward: During training, the agent is rewarded by a human referee who observes the distorted image and the image obtained after the transformation. The human gives a reward of $+1$ if the image obtained is *satisfactory enough* and a reward of -1 otherwise. The notion of a *satisfactory* image is quite ambiguous and depends on the person evaluating. This difference may result in an agent which is not robust enough. Thus, we train the agent with four different human referees by changing the referee after every 200 episodes. An image is termed to be *satisfactory* if an *average* person can identify the objects in the image with a certain degree of confidence.

3.2 ObjectRL

3.2.1 Problem Formulation

We cast the problem of image parameter modifications as a Markov Decision Process (MDP) [Puterman (1994)] since this setting provides a formal framework to model an agent that makes a sequence of decisions. Our formulation considers a single image as the state. To simulate the effect of *bad* images as well as increase the variance in the images in a dataset, we digitally distort the images. These digital distortions are carried out by randomly choosing α for a particular type of distortion (brightness, contrast, color). We have a pre-trained object detection network which could be trained either on the same dataset or any other dataset.

3.2.2 MDP for ObjectRL

States: The states for the agent are $128 \times 128 \times 3$ RGB images from the PascalVOC dataset Everingham *et al.* (2015) which are distorted by random factors α chosen according to the scale of distortion. We consider only one type of distortion (brightness, color, contrast) at a time, ie. we train different models for different types of the distortion. Combining all the different types of distortions in a single model remains to be a key direction to explore in future work.

Scales of Distortion: We perform experiments with the following two degrees of distortion in the image:

- Full-scale distortion: The random distortion in the images $\alpha \in [0, 2]$.
- Minor-scale distortion: The random distortion in the images $\alpha \in [0.5, 1.8]$. This constraint limits the images to not have distortions which cannot be reverted back with the action space, the agent has access to.

The variation of the the distorted images can be seen in Fig 2.2, 2.3, 2.4.

Actions: The agent can choose to change the global parameter (brightness, color, contrast) of the image by giving out a scalar $a_t \in [0, 2]$. Here, a_t is equivalent to α in the image distortion equations described in Section 2.3.5. The action a_t can be applied sequentially upto n number of times. After n steps the episode is terminated. Here, we set the value of $n = 1$ to achieve stability in training as having larger horizons lead to the images getting distorted beyond repair during the initial stages of learning and hence does not explore with the *better* actions.

Reward: First, we evaluate scores d_t for the images as follows:

$$d_t(x) = \gamma(IoU(x)) + (1 - \gamma)(F1(x)) \tag{3.1}$$

x is the input image to the pre-trained object detector. IoU is the average of all the intersection over union for the bounding boxes predicted in the image and F1 is the F1-score for the image. We set $\gamma = 0.1$ because we want to give more importance to the number of correct objects being detected.

We evaluate:

- $d_{o,t} = d_t(\text{original image})$
- $d_{d,t} = d_t(\text{distorted image})$
- $d_{s,t} = d_t(\text{state})$

where the *original image* is the one before the random distortion, *distorted image* is the image after the random distortion and *state* is the image obtained after taking the action proposed by the agent.

We define,

$$\beta_t = 2d_{s,t} - d_{o,t} - d_{d,t} \tag{3.2}$$

Here, β_t is positive if and only if the agent's action leads to an image which gives better detection performance than both the original image as well as the distorted image. Thus we give the reward (r_t) as follows:

$$r_t = \begin{cases} +1, & \text{if } \beta_t \geq -\epsilon \\ -1, & \text{otherwise} \end{cases}$$

Note that $d_{o,t}$ and $d_{d,t}$ do not change in an episode and only $d_{s,t}$ changes over the episode. We set the hyperparameter $\epsilon = 0.01$ as we do not want to penalise the minor shifts in bounding boxes which result in small changes in IoU in Eqn[3.1]. Fig 3.1 shows the training procedure for *ObjectRL*.

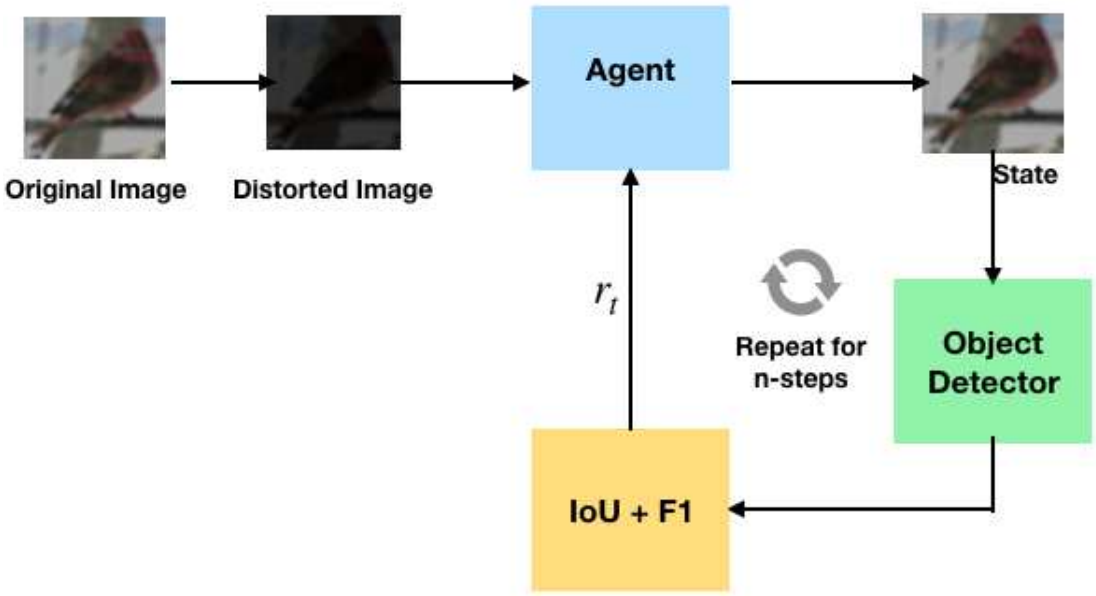


Figure 3.1: The overall training procedure for *ObjectRL*. The image is randomly distorted to simulate the bad images. An episode can be carried out for n steps which we set to 1 for training stability. Thus, the agent has to take a single action on each image.

3.3 Motivation for *ObjectRL*

In scenarios where object-detection algorithms are deployed in real-time, for example in autonomous vehicles or drones, lighting conditions and subject speeds can change quickly. If cameras use a single operation mode, the image might be quite blurred or dark and hence the image obtained may not be ideal for performing object detection. In

these cases it would not be possible to create new datasets with images obtained from all the possible combinations of camera parameters along with manually annotating them with bounding-boxes. Also, due to the lack of these annotated images we cannot fine-tune the existing object-detection networks on the distorted images. Our model leverages digital distortions on existing datasets with annotations to learn a policy such that it can tackle changes in image parameters in real-time to improve the object detection performance.

One of the main motivations of *ObjectRL* is to extend it to control camera parameters to capture images which are good for object detection in real time. Thus, we propose an extension to *ObjectRL* (for future work) where we have an RL agent which initially captures images by choosing random combinations of camera parameters (exploration phase). A human would then give rewards according to the objects detected in the images in the current buffer. These rewards would then be used to update the policy to improve the choice of camera parameters. This method of assigning a $\{\pm 1\}$ reward is comparatively much faster than annotating the objects in the image to extend the dataset and training a supervised model with this extended model. This methodology is quite similar to the DAgger method (Dataset Aggregation) by Ross *et al.* (2010) where a human labels the actions in the newly acquired data before adding it into the experience for imitation learning.

CHAPTER 4

EVALUATION

4.1 Experimental Setup

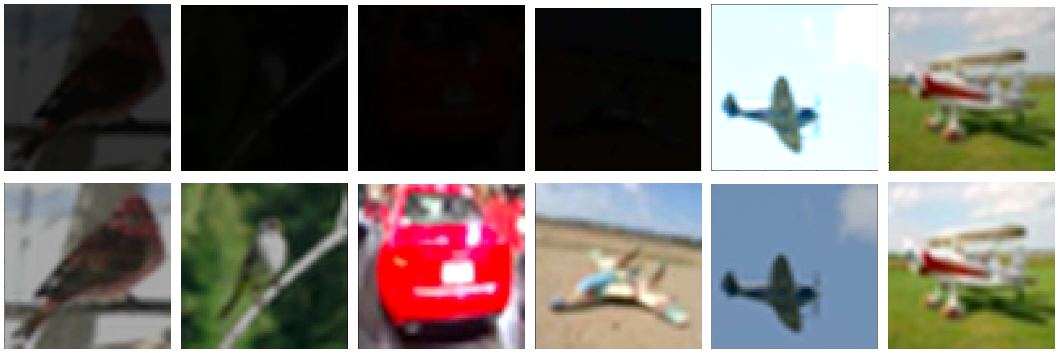
We have built our network with PyTorch. For the object detector, we use a Single Shot Detector (SSD) [Liu *et al.* (2015)] and YOLO-v3 [Redmon *et al.* (2015)] trained on the PascalVOC dataset with a VGG-base network [Simonyan and Zisserman (2014)] for SSD. We use Proximal Policy Optimization (PPO) [Schulman *et al.* (2017)] for optimising the *ObjectRL* agent. We train the agent network on a single NVIDIA GTX 1080Ti with the PascalVOC dataset.

For *hImgRL*, the agent network consists of 6 convolutional layers each with kernel size=5, stride=1 and padding=2. The number of filters are {32, 64, 64, 64, 64, 64}. After every two layers we have a maxpooling layer with kernel size=2 and stride=2, followed by a ReLU activation function. After the convolutional layers, there is one hidden linear layer with 512 nodes and an output layer with 41 nodes. Each of the 41 nodes correspond to an action from the set $\mathcal{A}$ as described in section 3.1.2. The agent is updated after 20 episodes for 2 epochs with batch-size=10.

For *ObjectRL*, both the actor and the critic networks consist of 6 convolutional layers with (kernel size, stride, number of filters)= {(4,2,8), (3,2,16), (3,2,32), (3,2,64), (3,1,128), (3,1,256)} followed by linear layers with output size 100, 25, 1. The agent is updated after 2000 episodes for 20 epochs with batch-size=64.

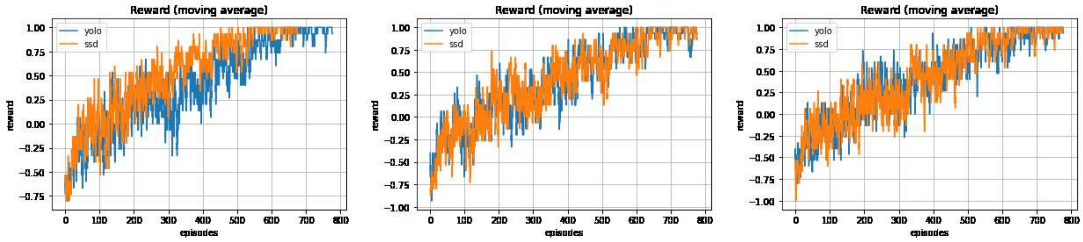
The learning rate for both *hImgRL* and *ObjectRL* agents is 10^{-3}. We use Adam Optimizer [Kingma and Ba (2014)] which is a default optimizer provided in PyTorch.

In the initial episodes, for both, *hImgRL* and *ObjectRL* we take actions in an $\epsilon-$*Greedy* manner. ϵ is annealed linearly with the number of episodes until it reaches 0.05.



(a) (b) (c) (d) (e) (f)

Figure 4.1: Actions taken by *hImgRL* on test data. The top row contains images given to the agent. The bottom row contains images after applying the transformations proposed by the agent. All the above transformations made by the agent are with episode horizon=1.



Brightness Color Contrast

Figure 4.2: Episodic return of the *ObjectRL* while training with a moving average of size 30. Each iteration represents 1K episodes.

4.2 Results

4.2.1 hImgRL

In Figure 4.1, we show a few of the outputs of the agent on test data, after training. We terminate the training of the agent when the average reward over previous 200 images is 200 i.e. it recovers the image *satisfactorily* for 200 consecutive images after the exploration phase is over. The first four columns (a-d) in Figure 4.1 are examples of dark images getting transformed to a brighter image. Column (e) is an example of a bright image getting transformed to a slightly darker one. Column (f) is an example of an *ideal* image not changing much after the agent's transformation.

k	Brightness							
	Full-scale				Minor-scale			
	SSD		YOLO		SSD		YOLO	
	GS	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ours*
1	955 ± 14	532 ± 20	1360 ± 22	976 ± 18	435 ± 25	428 ± 23	1025 ± 23	883 ± 24
2	154 ± 6	87 ± 3	202 ± 15	118 ± 15	87 ± 12	80 ± 9	85 ± 15	63 ± 15
3	49 ± 3	32 ± 4	52 ± 8	18 ± 6	14 ± 5	12 ± 3	8 ± 2	5 ± 1
4	18 ± 3	7 ± 1	17 ± 2	4 ± 1	5 ± 1	3 ± 0	2 ± 0	0
5	7 ± 2	2 ± 0	4 ± 1	2 ± 0	0	0	0	0

Table 4.1: *TP-Score(k)* with brightness distortion. GS stands for Grid-Search.

4.2.2 Measure for evaluation for ObjectRL: TP-Score

To the best of our knowledge, we believe no suitable measure is defined for this problem and hence we define a measure called *TP-Score(k)* (True Positive Score). This score is the number of images in which $k-$or more true positives were detected which were not detected in the image before transformation. The *TP-Score(k)* is initialised to zero for a set of images $\mathcal{I}$. For example: Let the number of true-positives detected before the transformation be 3 and let the number of true-positives detected after the transformation be 5. Then we have one image where 2 extra true-positives were detected which were not detected in the input image. Thus, we increase *TP-Score(1)* and *TP-Score(2)* by one.

4.2.3 Baseline for ObjectRL

To obtain the baselines, we first distort the images in the original dataset. The images are distorted with α being randomly chosen from the set $\mathcal{S} = \{0.1, \ldots, 1.9, 2.0\}$ or $\mathcal{S} = \{0.5, \ldots, 1.7, 1.8\}$ depending on the scale. The set of available actions to be applied on on these images are: $\hat{\mathcal{S}} = \{\frac{1}{s} \forall s \in \mathcal{S}\}$. We evaluate the *TP-Score(k)* on the distorted images by applying the transformations by performing a grid-search over all $\alpha \in \hat{\mathcal{S}}$ and report the scores obtained with the best-performing actions for different types and scales of distortions in Table 4.1, 4.2 and 4.3. We also report the *TP-Scores* obtained after applying the transformations proposed by *ObjectRL* on the images distorted using full-scale and minor-scales. The scores reported are averaged over 10 image sets $\mathcal{I}$, each containing 10,000 images. Note that the means and standard deviations are rounded to the nearest integers.

k	Color							
	Full-scale				Minor-scale			
	SSD		YOLO		SSD		YOLO	
	GS	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ours*
1	973 ± 17	672 ± 19	1250 ± 23	1103 ± 21	561 ± 18	532 ± 22	974 ± 21	930 ± 22
2	123 ± 7	84 ± 4	210 ± 16	135 ± 13	43 ± 9	37 ± 9	83 ± 12	82 ± 12
3	53 ± 4	31 ± 3	63 ± 7	23 ± 6	1 ± 0	0	15 ± 2	10 ± 1
4	11 ± 2	3 ± 1	19 ± 2	5 ± 1	0	0	6 ± 1	3 ± 0
5	5 ± 1	1 ± 0	6 ± 1	2 ± 0	0	0	0	0

Table 4.2: *TP-Score(k)* with color distortion. GS stands for Grid-Search.

k	Contrast							
	Full-scale				Minor-scale			
	SSD		YOLO		SSD		YOLO	
	GS	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ObjectRL*	*GS*	*ours*
1	955 ± 15	532 ± 20	1360 ± 21	976 ± 19	680 ± 22	663 ± 24	1038 ± 23	975 ± 24
2	163 ± 8	101 ± 4	213 ± 16	134 ± 15	62 ± 10	49 ± 9	104 ± 13	85 ± 15
3	55 ± 4	36 ± 4	67 ± 7	39 ± 6	14 ± 3	6 ± 2	19 ± 3	16 ± 2
4	21 ± 2	11 ± 1	28 ± 2	13 ± 1	1 ± 0	1 ± 0	5 ± 0	3 ± 0
5	4 ± 1	2 ± 0	5 ± 1	2 ± 0	0	0	0	0

Table 4.3: *TP-Score(k)* with contrast distortion. GS stands for Grid-Search.

As seen in Table 4.1,4.2 and 4.3, *ObjectRL* is not able to perform as well as the grid-search for full-scale distortions. The reason for this is that many of the images obtained after the full-scale distortions are not repairable with the action set provided to the agent.

But with minor-scale distortions, *ObjectRL* is able to perform as well as the grid-search. The total time taken for the grid-search over all brightness values for one image is 12.5094 ± 0.4103s for YOLO and 15.1090 ± 0.3623 for SSD on a CPU. The advantage of using *ObjectRL* is that the time taken by the agent is 10 times less than grid-search. This latency is quite crucial in applications like surveillance drones and robots where the lighting conditions can vary quickly and the tolerance for errors in object-detection is low.

4.2.4 Discussion on the outputs of *ObjectRL*

In this section, we discuss the outputs obtained from *ObjectRL* with SSD and minor-scale distortion which are shown in Fig 4.3. In column (a) 4 true positives are detected

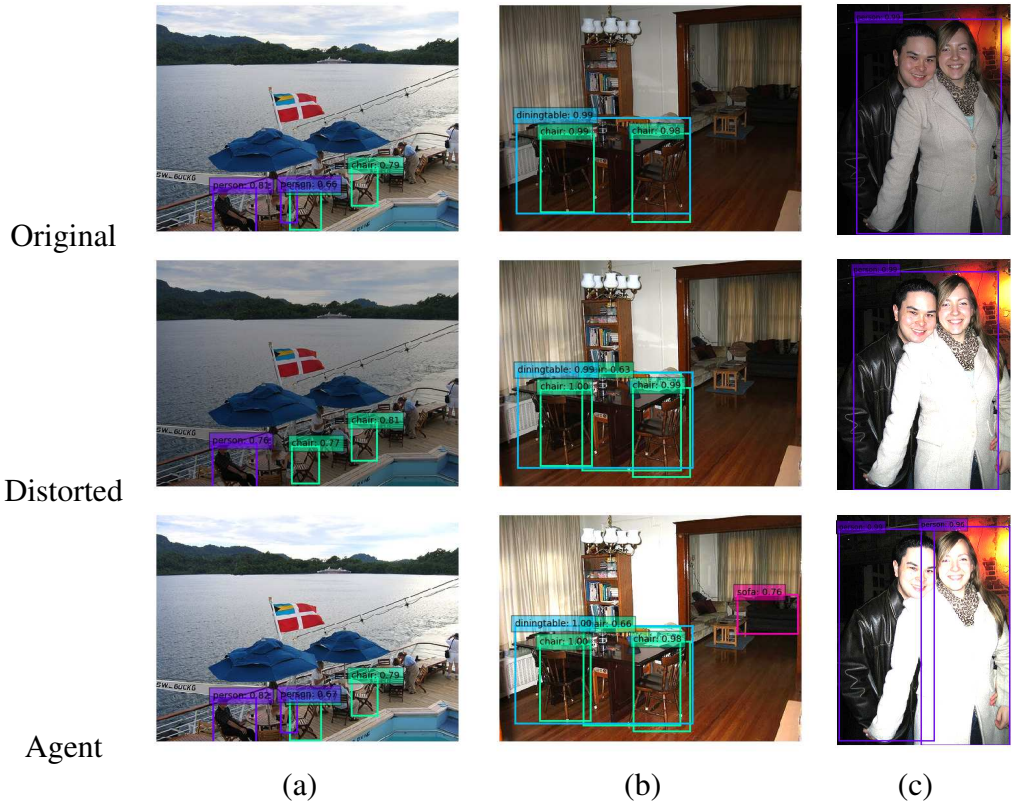


Figure 4.3: A few of the outputs from *ObjectRL* with SSD and minor-scale distortion. The top row contains the original images. The second row contains the distorted images. The bottom row contains images obtained from the agent. Bounding boxes are drawn over the objects detected by the detector.

in the original image, 3 true positives are detected in the distorted image and 4 true positives are detected in the original image. The distorted image is slightly darker the the original one. *ObjectRL* is able to recover the object lost after distortion. In column (b) 3 true positives are detected in the original image, 4 true positives are detected in the distorted image and 5 true positives are detected in the original image. In this case, even the distorted image performs better than original image. But the agent-obtained image performs the best with 5 true-positives. In column (c) 1 true positive is detected in the original image, 1 true positive is detected in the distorted image and 2 true positives are detected in the original image. In this case the agent obtained image outperforms both the distorted and the original image. For a human eye, the agent-obtained image may not look *pleasing* as it is much brighter than the original image. Ideally for a human, the distorted image in column (c) is the most *pleasing*. Column (c) is one of the perfect examples to demonstrate the fact that whatever looks pleasing to a human eye may not necessarily be the optimal one for object-detection. Thus on an average, the agent is able to recover either as many objects as detected in the original image or more.

k	Brightness		Color		Contrast	
	π_{yolo}^{ssd}	π_{ssd}^{yolo}	π_{yolo}^{ssd}	π_{ssd}^{yolo}	π_{yolo}^{ssd}	π_{ssd}^{yolo}
1	582 ± 13	1045 ± 24	800 ± 15	1249 ± 26	813 ± 15	1243 ± 26
2	36 ± 6	73 ± 11	72 ± 8	138 ± 11	65 ± 8	145 ± 12
3	2 ± 0	9 ± 4	10 ± 1	13 ± 3	2 ± 0	19 ± 4

Table 4.4: *TP-Score(k)* by crossing the policies.

According to our experiments, there were 8 ± 1 images with SSD and 34 ± 5 images with YOLO-v3, where the agent-obtained image had lesser number of true-positives than the original image. Although, this number of true-positives was more than the number of true-positives detected in the distorted image.

4.2.5 Crossing Policies

In this section we perform experiments by swapping the detectors for the learned policies. Thus, we use π_{yolo} with SSD, (denoted as π_{yolo}^{ssd}) and π_{ssd} with YOLO, (denoted as π_{ssd}^{yolo}). In Table 4.4, we report the number of images where $k-$or lesser true positives were detected with the swapped policy than what were detected using the original policy on their corresponding detectors. As shown in Table 4.4, π_{SSD} on YOLO is worse than π_{YOLO} on SSD. This is because the range of values for which SSD gives optimal performance is bigger than the range of values for which YOLO gives optimal performance. In essence, YOLO is more sensitive to the image parameters than SSD.

CHAPTER 5

CONCLUSION

This work proposes the usage of reinforcement learning to improve the object detection of a pre-trained object detector network by changing the image parameters (*ObjectRL*). We validate our approach by experimenting with distorted images and making the agent output actions necessary to improve detection. Our experiments showed that pre-processing of images is necessary to extract the maximum performance from a pre-trained detector. Future work includes combining all the different distortions in a single model and using it for controlling camera parameters to obtain images. Along with this, local image manipulations such as changing the image parameters only in certain regions of the image could be tried out.

REFERENCES

1. **Andreopoulos, A.** and **J. K. Tsotsos** (2012). On sensor bias in experimental methods for comparing interest-point, saliency, and recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**(1), 110–126.

2. **Bychkovsky, V.**, **S. Paris**, **E. Chan**, and **F. Durand**, Learning photographic global tonal adjustment with a database of input / output image pairs. *In CVPR 2011*. 2011. ISSN 1063-6919.

3. **Caicedo, J. C.** and **S. Lazebnik** (2015). Active object localization with deep reinforcement learning. *CoRR*, **abs/1511.06015**. URL `http://arxiv.org/abs/1511.06015`.

4. **Everingham, M.**, **S. M. A. Eslami**, **L. Van Gool**, **C. K. I. Williams**, **J. Winn**, and **A. Zisserman** (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, **111**(1), 98–136.

5. **Goodfellow, I. J.**, **J. Pouget-Abadie**, **M. Mirza**, **B. Xu**, **D. Warde-Farley**, **S. Ozair**, **A. Courville**, and **Y. Bengio**, Generative adversarial nets. *In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14. MIT Press, Cambridge, MA, USA, 2014.

6. **Hasinoff, S. W.**, **F. Durand**, and **W. T. Freeman**, Noise-optimal capture for high dynamic range photography. *In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010. ISSN 1063-6919.

7. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15. IEEE Computer Society, USA, 2015*a*. ISBN 9781467383912. URL `https://doi.org/10.1109/ICCV.2015.123`.

8. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15. IEEE Computer Society, USA, 2015*b*. ISBN 9781467383912. URL `https://doi.org/10.1109/ICCV.2015.123`.

9. **Hu, Y.**, **H. He**, **C. Xu**, **B. Wang**, and **S. Lin** (2018). Exposure: A white-box photo post-processing framework. *ACM Trans. Graph.*, **37**(2). ISSN 0730-0301. URL `https://doi.org/10.1145/3181974`.

10. **Kingma, D. P.** and **J. Ba** (2014). Adam: A method for stochastic optimization. URL `http://arxiv.org/abs/1412.6980`. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

11. **Koenig, J.**, **S. Malberg**, **M. Martens**, **S. Niehaus**, **A. Krohn-Grimberghe**, and **A. Ramaswamy** (2018). Multi-stage reinforcement learning for object detection. *CoRR*, **abs/1810.10325**. URL http://arxiv.org/abs/1810.10325.

12. **Konda, V. R.** and **J. N. Tsitsiklis**, Actor-critic algorithms. *In* **S. A. Solla**, **T. K. Leen**, and **K. Müller** (eds.), *Advances in Neural Information Processing Systems 12*. MIT Press, 2000, 1008–1014. URL http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf.

13. **Krizhevsky, A.** (2009). Learning multiple layers of features from tiny images. Technical report.

14. **Linderoth, M.**, **A. Robertsson**, and **R. Johansson**, Color-based detection robust to varying illumination spectrum. *In 2013 IEEE Workshop on Robot Vision (WORV)*. 2013.

15. **Liu, W.**, **D. Anguelov**, **D. Erhan**, **C. Szegedy**, **S. E. Reed**, **C. Fu**, and **A. C. Berg** (2015). SSD: single shot multibox detector. *CoRR*, **abs/1512.02325**. URL http://arxiv.org/abs/1512.02325.

16. **Maier, W.**, **M. Eschey**, and **E. Steinbach**, Image-based object detection under varying illumination in environments with specular surfaces. *In 2011 18th IEEE International Conference on Image Processing*. 2011.

17. **Mathe, S.**, **A. Pirinen**, and **C. Sminchisescu**, Reinforcement learning for visual object detection. *In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

18. **Mathe, S.** and **C. Sminchisescu** (2014). Multiple instance reinforcement learning for efficient weakly-supervised detection in images. *CoRR*, **abs/1412.0100**. URL http://arxiv.org/abs/1412.0100.

19. **Osadchy, M.** and **D. Keren**, Image detection under varying illumination and pose. *In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2. 2001.

20. **Osadchy, M.** and **D. Keren** (2004). Efficient detection under varying illumination conditions and image plane rotations. *Computer Vision and Image Understanding*, **93**, 245–259.

21. **Park, J.**, **J. Lee**, **D. Yoo**, and **I. S. Kweon** (2018). Distort-and-recover: Color enhancement using deep reinforcement learning. *CoRR*, **abs/1804.04450**. URL http://arxiv.org/abs/1804.04450.

22. **Puterman, M. L.**, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994, 1st edition. ISBN 0471619779.

23. **Redmon, J.**, **S. K. Divvala**, **R. B. Girshick**, and **A. Farhadi** (2015). You only look once: Unified, real-time object detection. *CoRR*, **abs/1506.02640**. URL http://arxiv.org/abs/1506.02640.

24. **Ross, S.**, **G. J. Gordon**, and **J. A. Bagnell** (2010). No-regret reductions for imitation learning and structured prediction. *CoRR*, **abs/1011.0686**. URL http://arxiv.org/abs/1011.0686.

25. **Schulman, J.**, **F. Wolski**, **P. Dhariwal**, **A. Radford**, and **O. Klimov** (2017). Proximal policy optimization algorithms. *CoRR*, **abs/1707.06347**. URL http://arxiv.org/abs/1707.06347.

26. **Simonyan, K.** and **A. Zisserman** (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, **abs/1409.1556**. URL http://arxiv.org/abs/1409.1556.

27. **Williams, R. J.** (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, **8**(3–4), 229–256. ISSN 0885-6125. URL https://doi.org/10.1007/BF00992696.

28. **Wu, Y.** and **J. K. Tsotsos** (2017). Active control of camera parameters for object detection algorithms. *CoRR*, **abs/1705.05685**. URL http://arxiv.org/abs/1705.05685.

29. **Yang, H.**, **B. Wang**, **N. Vesdapunt**, **M. Guo**, and **S. B. Kang** (2018). Personalized attention-aware exposure control using reinforcement learning. *CoRR*, **abs/1803.02269**. URL http://arxiv.org/abs/1803.02269.