Object Detection: A Reinforcement Learning approach using Adversarial Models

Angel A. Cantu  
*Department of Computer Science*  
*University of Texas Rio Grande Valley*  
Edinburg, United States of America  
angel.cantu01@utrgv.edu

Nami Akazawa  
*Department of Computer Science*  
*University of Texas Rio Grande Valley*  
Edinburg, United States of America  
nami.akazawa01@utrgv.edu

Abstract—In this research experiment we attempt to remove the need of a human to label objects in images as a pre-processing procedure for object detection algorithms. We come up with a model that uses a reinforcement learning agent and an artificial neural network to form an adversarial network. This network is used to construct a reward system for the agent as the agent performs. The network will learn which actions are best for the agent as the network learns the similarities of some dataset and the agent’s output. The agent performs in a dataset of images, where it chooses what dimensions to crop from the image. For now, we consider a fixed bounding box of size $80 \times 80$. This cropped image is sent to the network for evaluation, where the network will determine if it is similar to some object.

Index Terms—Machine Learning, Adversarial Networks, Object Detection, Reinforcement Learning

I. INTRODUCTION

Object detection is a computer technology related to computer vision that deals with the task of detecting instances of some particular object in digital images and videos. Common uses are facial detection, image retrieval, video surveillance and many more. The most common methods of achieving functional object detection generally fall within machine learning-based algorithms and deep learning. One of the most popular deep learning algorithms is that of You Only Look Once (YOLO). YOLO’s procedure is to use a neural network that divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. Figure 1 depicts YOLO’s object detection algorithm in detail. Although YOLO has great performance, there is an expensive pre-processing method that is needed in order to prepare some dataset for training. This pre-processing requires the researcher to label the objects in the images using a free software distributed by YOLO. For small datasets this is easily dealt with; with large datasets, however, is a long process that usually requires the combined work of many researchers. The free software offered by YOLO is shown in Figure 2. This software is available at their GitHub repository. In this software, you select some folder which contains all your images. You use the buttons in the software to select images one-by-one and use the selection tool to decide the dimensions of the bounding box. This produces some `.txt` file with the dimensions and the image’s file name. This information is fed into YOLO’s network.

![Figure 1: YOLO’s object detection examples.](image-url)
II. Motivation

As our technology advances over the years, there exists some areas of advancement where human interaction is necessary. Making accurate pre-processing often requires a human to be involved in the procedure. We take the first steps in solving the problem of automatization in data pre-processing. The goal of this research is to automatize the process of image labeling by using a reinforcement learning agent in an adversarial environment. We aim to speed up the advancement of technological experiments with the use of intelligent agents in order to save and create time for humans.

III. Methods

To address the problem of human labeling, we use a state-of-the-art machine learning-based algorithm called Generative Adversarial Networks (GANs). GANs are a class of machine learning systems that puts two neural networks to contest against each other, forming a zero-sum game framework. The system consists of two networks called the generator and discriminator. The generator network’s generates candidates while the discriminator network evaluates them. This zero-sum game operate in terms of data distributions; typically, the generator learns to form a mapping function from latent space to the data distribution of the real dataset. The real dataset consists of the datasets which the generator will essentially learn to generate. These generated datas are send to the discriminator for evaluation. If the generator “fools” the discriminator, the generator’s error rate decreases while the discriminator’s increases. To fool the discriminator is the discriminator classifying the generated data as real.

The goal of the agent is to receive an image and produce a center point where the bounding box should be placed. For now, we consider only a bounding box of size $80 \times 80$. Formally, the agent’s details are:

- Environment: The agent’s environment is best described as the images that require annotation.
- Actions: The agent’s actions consists of giving two integers that dictate the center location of the bounding box. These two integers are less than or equal to the size of the image.
- Observation: The observation after each action is the image used as the environment with the bounding box placed in the image. We do this to allow the agent to see where the bounding box was placed.
- Reward: The discriminator uses binary classification to detect which images are real or fake. Real images are classified as [1] and fake as [0]. After the agent sends the bounding box image to the discriminator, the agent uses the prediction as its reward. If the discriminator detects the agent’s cropped image as real, then the prediction will be of higher value (closer to 1).

By using an Agent as our generator in a GAN system we do not only automatize the labeling part for object detection but create an agent that is able to detect objects itself. The environment is depicted in figure 4.
at which locations does the object exist in the image. Our environments have dimensions $225 \times 300$. We also considered converting the images to black and white. After the agent produces two integers of which are used to create some bounding box, we send those integers and the environment (image) through some piece of code which removes that particular area from the image. Figure 5 depicts this process. Immediately after, we send the cropped image to the discriminator for evaluation.

The discriminator has access to the real dataset, which consists of ground-truth images of the particular object. The discriminator essentially is what dictates how good the agent may perform. We therefore consider different methods to training the discriminator:

- We consider pre-training the discriminator before using it for evaluations. There is one problem though: although we have the ground-truth for the real datasets, we do not have access to some ground-truth for the non-object data. We have attempted pre-training the discriminator on the real data and on random crops from the discriminator’s data, though the reward system does not seem to work well this way.
- We also consider training the discriminator concurrently with the agent. This allows for both actors to maintain an equal level of performance. We currently use this method to test our model.

![Figure 5: The agent performs and crops some fixed space from the image.](image)

![Figure 6: The cropped image is send to the discriminator for validation.](image)

![Figure 7: The discriminator produces some prediction value $x$ which is used for the agent’s reward system.](image)

![Figure 8: Real dataset that the discriminator uses to make evaluations.](image)

![Figure 9: Fake dataset that the agent uses to make bounding boxes.](image)

IV. DATA

We used many images from online resources. Most of these resources were from universities that offer free face datasets online. We also use OpenAI’s baselines algorithms. We currently test our method using PPO2.

![Figure 10: A single star in a white background, as depicted in Figure 10. As we let our model train longer time, our agent learns to put a bounding box in a correct position. Once we saw a good result from our first evaluation, we decided to perform face detection using a face](image)

V. RESULTS

For our first evaluation, we use a simple environment: A single star in a white background, as depicted in Figure 10. As we let our model train longer time, our agent learns to put a bounding box in a correct position. Once we saw a good result from our first evaluation, we decided to perform face detection using a face
Figure 10: Simple environment consisting of a single star.

dataset that is publicly available online. This increases the difficulty for our model to learn since every face images are under various conditions such as illumination, background, face alignment and facial expression. This required a longer training than the first experiment for us to see some good results. Despite the fact that our fixed bounding box size $80 \times 80$ is not large enough to cover the whole face in the image, as we keep training our model, it was able to crop. In Figure 11, we see

Figure 11: Two examples of the agent’s performance.

the results after a few hours of training. However, there were still many misplacements of bounding boxes on the faces. We suspect that the requirement of a fixed bounding box limits the agent to what it is capable of performing. For our future work, we will allow the agent to choose a larger bounding box, along with multiple bounding boxes.

ACKNOWLEDGMENT

We like to thank Adolfo Gonzalez III for helping us with experimental questions and idea sharing. We also like to thank Dr. Kim for the help and advice towards the current state of this project.

REFERENCES
