



# Bangladesh University of Business and Technology

# BUBT

Committed to Academic Excellence

Project Name.

## Dino Gaming Bot

**Course Title: Artificial Intelligence Lab**

**Code: CSE 319**

Submitted To

Dr. M. Firoz Mridha

Associate Professor, Dept. of CSE

Submitted By		
Name	ID	Program: B.Sc.Engg. in CSE Intake: 36 Section: 01
MD. Ashrafuzzaman	ID:16173103136	
MD. Nazirul Islam	ID:16173103041	

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	DECLARATION	iv
	DEDICATION	v
	ACKNOWLEDGMENT	vi
	ABSTRACT	vii
	TABLE OF CONTENTS	vii
	LIST OF TABLES	xii
	LIST OF FIGURES	xiii
<b>1</b>	<b>INTRODUCTION</b>	
1.1	Introduction	
1.2	Problem Background	
1.3	Project Objectives	
1.4	Motivations of the Project	
1.5	Project Contributions	
1.6	Organization of the Project Report	
<b>2</b>	<b>BACKGROUND</b>	
2.1	Existing Systems	
2.2	Problem Analysis	
2.3	Supporting Theory	
<b>3</b>	<b>PROPOSED MODEL</b>	
3.1	Feasibility Analysis	
3.2	Requirement Analysis	
3.3	System Architecture	
3.4	System Design	
3.5	Implementation	
<b>4</b>	<b>IMPLEMENTATION AND TESTING</b>	
4.2	Result Analysis	
4.3	Application Outcome	
<b>5</b>	<b>CONSTRAINTS AND ALTERNATIVES</b>	
6.1	Design Constraints	
6.2	Component Constraints	
<b>6</b>	<b>CONCLUSION</b>	
6.1	Introduction	
6.2	Future Works	

## **DECLARATION**

We declare that this project and the work presented in it is our own and has been generated by us as the result of our own original research. Sometimes I got help from our honorable teacher and we also got help from internet.

Finally, we confirm that:

- ✚ This work is done wholly or mainly while in candidature for a research degree at this University.
- ✚ This project work has not been previously submitted for any degree at this university or any other educational institutes of Bangladesh.
- ✚ I have quoted from the work of others; the source is always given respectively. With the exception of such quotations.

## **DEDICATION**

Dedicate to our parents and siblings for all their love, support and inspiration.

## ACKNOWLEDGEMENTS

Any attempt at any level cannot be satisfactorily completed without the support and guidance of learned people. We would like to express our immense gratitude to all BUBT teachers on computer programming for their constant support and motivation that has encouraged us to come up with this project.

We are profoundly grateful to **Dr. M. Firoz Mridha** his expert guidance and continuous encouragement throughout to see that this project rifts its target since its commencement to its completion.

We would like to express deepest appreciation towards **Wing Commander Md Momenul Islam Retd**, proctor, Bangladesh University of Business & Technology, **Prof. Dr. Md. Ameer Ali** head of Department of Computer Science & Engineering whose invaluable guidance supported us in completing this project.

## **ABSTRACT**

In this project, we implement both feature-extraction based algorithms and an end-to-end deep reinforcement learning method to learn to control Chrome offline dinosaur game directly from high-dimensional game screen input. Results show that compared with the pixel feature based algorithms, deep reinforcement learning is more powerful and effective. It leverages the high-dimensional sensory input directly and avoids potential errors in feature extraction. Finally, we propose special training methods to tackle class imbalance problems caused by the increase in game velocity. After training, our Deep-Q AI is able to outperform human experts.

### **Keywords**

Deep Q-Learning; MLP; Feature Extraction with OpenCV

# Chapter 1 Introduction

---

## *1.1 Introduction*

T-Rex run game is a game which we are sure every human with a phone or a laptop with Google Chrome and no internet connection has played. For those of you who do not know, just turn off your internet connection on your phone or laptop and open any website using Google Chrome and you'll be able to play T-Rex run! The basic objective of the game is to jump or duck and dodge the obstacles and keep your Dino alive. However, being humans we might get bored or might get distracted and probably loose the game. In order to beat this game to the fullest, we require someone with extreme concentration and someone who cannot get distracted with anything in this world, someone like a BOT!

## *1.2 Problem Background*

### ***1.3 Project Objectives***

Dino Run Game project is composed purely in Python. The task documents consist of image documents as well as a python script (main.py). GUI utilizes the pygame library. Speaking about the gameplay, it is a cloned version of an offline game played in google chrome calling “T-Rex Dino Run”. The main objective of this mini-game is to rack up increasingly more point without being touched by any kind of challenges. All the playing methods coincide. The customer needs to play this straightforward game utilizing two Keyboard keys. Spacebar to leap as well as Down arrowhead key to hide. In contrast to the initial one, this clone variation has some modifications in the video gaming environment

### ***1.4 Motivations of the Project***

### ***1.5. Project Contributions***

### ***1.6 Organization of the Project Report***



# Chapter 2 Background

---

## 2.1 Existing Systems

## 2.2 Problem Analysis



The basic logic what we will be following here in building the bot is to mimic how a human plays the game. A human sees the obstacle and tries to time the jump of the Dino to avoid the obstacle. For our bot, we will be doing the same. Basically, here we take a screenshot of the T-Rex game screen and check the position of a tree or a bird obstacle in that image and if the obstacle is close enough to the Dino (defined using a threshold defined in the code), we ask the Dino to jump.

The first step is to take the screen shot of the web page with the game and locate the exact pixel values of the Dino. We find the top and bottom corner pixel values of the Dino. This can be done by importing the image into paint.net and finding the pixel values. Also, if you notice, the position of the Dino is constant and only the obstacles are moving towards it. Hence we take this approach of creating an action on the Dino based on its static position with respect to the obstacles moving forward towards it. The below figure shows how we can get the pixel positions of the Dino figure, using the rulers and the grid view. Usually the sum of all pixel values in an image adds up to a number, and if the image does not change, the sum always remains constant. We keep this in mind and I'll explain why this is important in the next few paragraphs.

Similarly, we take a screenshot of the game when a tree is present, a bird is present and replay button is present and note down its pixel locations. This is not much required as of now, but in the future can be helpful

## 2.3 Supporting Theory

I will be building this bot in python using image processing libraries. To start with, you will be required to install Python 3.6 or above on your PC or laptop. Here are the list of libraries you'll be required to install —

-  pyautogui — This is a library which has functions to interact with the T-Rex run GUI
-  PIL — PIL Library is used for taking screenshots of the screen and applying image processing on it

- ✚ Time — Used for giving different delays between jumps
- ✚ ImageGrab — ImageGrab is a Python module that helps to capture the contents of the system.
- ✚ ImageOps — The ImageOps module contains a number of ‘ready-mode’ image processing operations. This module is somewhat experimental and most operators only work on L and RGB image.
- ✚ NumPy — This article will help you get acquainted with the widely used array-processing library in Python, NumPy.

These libraries can be installed using the command “pip install <library name>”. Loads of documentation is available on w3schools on using pip and other commands and basics of working with python. Here is a link to it —

[https://www.w3schools.com/python/python\\_pip.asp](https://www.w3schools.com/python/python_pip.asp)

We then load our prerequisites into our code by the following snippet —

```
import time
from PIL import ImageGrab, ImageOps
import pyautogui
from numpy import *
```

## Chapter 3 Proposed Model

---

### *3.1 Feasibility Analysis*

### *3.2 Requirement Analysis*

### *3.3 System Architecture*

### *3.4 System Design*

### *3.5 Implementation*

There are 2 different objects in this game — Dino and replay button if you crash. You can use the pixel location values which I have used for each of these characters. I have used a class definition hold onto these values for making it simple. This is how to looks —

```
class Coordinates():
    replayBtn = (341,386)
    dinosaur = (370,390)
```

The next step is to actually write a logic for this code. We create different functions for different processes or objectives.

First we create a function which would capture every frame or take screen shot of every frame on the game in order to process it. Think it like how humans see the game while playing it, the bot also needs to see the game using screenshots. This is shown in the following snippet.

```
def imageGrab():

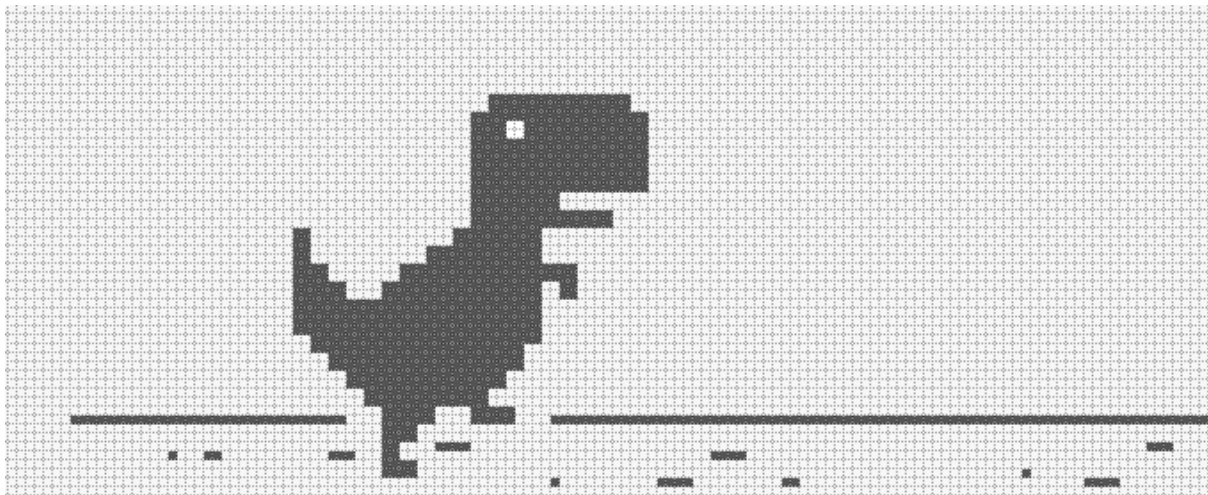
    box = (190, 365,
          280, 419)

    image = ImageGrab.grab(box)
    grayimage = ImageOps.grayscale(image)
    a = array(grayimage.getcolors())
    print(a.sum())
    return a.sum()
```

In the above function imagegrab(), we get the coordinates of the Dino and create a box around it. Basically the box is a rectangle with 4 pixel location values of each corner, with the Dino inside the box. The box has some open space on right side of the Dino. We then take a screen shot of the box and the box acts like our logic for creating a jump or duck action. The basic logic here is that if a tree or a bird comes inside the box, the Dino is asked to jump or duck respectively.

But bots actually cannot see what's happening in the screenshot or precisely inside the box. Bots can only see and read pixel. Hence, here as hinted in the previous section, we calculate the sum of all pixels within the box. As the image of the Dino and the box is constant and Dino is not moving, the sum of pixels in the box always returns a constant. But if a tree or bird enters the box, the sum

value actually increases or decreases based on the pixel values, and that is when the bot gets a hint to either jump or duck. The below image displaces the screenshot or how the box would look.



Once we have the image of the Dino and the box, we convert the box into gray scale to increase up our processing as color is not an important factor here as we are just bothered with calculating the sum of the pixel values in the box.

The next function we write is the jump function, which basically asks the bot to jump when a tree appears. Pyautogui library is used to simulate keyboard controls which enables the bot to interact with the game.

Similarly a duck function is written. You can read more about pyautogui here — <https://pyautogui.readthedocs.io/en/latest/keyboard.html>

```
def pressSpace():  
    pyautogui.keyDown('space')  
    print("jump")  
    time.sleep(0.18)  
    pyautogui.keyUp('space')
```

The next function we write is to restart the game. If by chance the bot fails (very low chance), it should automatically restart without human aid. We use the replay button coordinates and simulate a click on that location. Here is the code snippet for it.

```
def restartGame():  
    pyautogui.click(Cordinates.replayBtn)
```

Once done with all our functions, we are ready to write our main function and run it. The below snippet shows the main function.

```
def main():  
    restartGame()  
    while True:  
        if (imageGrab() != 5107):  
            pressSapce()  
            time.sleep(0.1)  
  
main()
```

## Chapter 4 Implementation and Testing

---

### *4.1 Result Analysis*

This is how the bot runs indefinitely and thus this is how we beat the T-Rex run game. This logic can be utilized in many different games too, that is the logic of using the summation of pixel values. We should always take into consideration that the bots only see pixels and nothing else. For future scope we can actually calculate and find the values of the locations of the tree and birds dynamically using OpenCV and other image processing techniques.

### *4.2 Application Outcome*

# Chapter 5 Constraints and Alternatives

---

*5.1 Design Constraints*

*5.2 Component Constraints*

## Chapter 6 Conclusion

---

### *6.1 Introduction*

Several approaches were used to achieve the AI that can play Chrome Offline Dinosaur Game. For the feature extraction based algorithm, computer vision methods can recognize the T-Rex and obstacles from the images. Carefully designed feature extraction algorithms can successfully abstract the state and AI built upon them can improve its performance significantly compared with naive baseline. MLP learned from online training can strengthen the AI further for it refines the parameters automatically by experience. However, feature-extraction based algorithm have their limits and cannot outperform the human experts. For end-to-end Deep-Q learning method, our result shows that it can successfully play the game by learning straightly from the pixels without feature extraction, and is much stronger than the feature-based method. Finally, specially designed training method can help us overcome the training difficulties caused by the properties of our game, which further improves our AI's performance and helps achieve super-human results.

### *6.2 Future Works*

In our project, neither MLP nor Deep Q-learning handles velocity well enough. In Deep Q-learning, we notice velocity makes great impact over the jumping position selection. During the process of training agent with different acceleration and velocity, we observe that the agent tends to use policies incorrectly fitting the current velocity when the relative speed changes. Two possible explanations are as follows. Firstly, we simplify the computation in neural networks by resizing the raw game image into  $80 \times 80$  pixels. In this way, the edge of the obstacles would be obscure which negatively influenced the prediction. Another problem is that we use four images in stacking to infer the relative velocity based on their differences. If more channels are added in the game image, more deviation would be detected in both MLP and Deep Q-learning in order to capture the change of velocity.