DINO AI GAME DESIGN DOCUMENT :

DATASET AND FEATURES :

We are implementing this game in python using the screenshots of various instances as our dataset .The environment is composed of a sliding screen composed of trees and flying birds and the dino agent needs to dodge them to survive in the game .
The speed of the screen keeps on increasing as time passes . After each generation of this game , our dinosaur agent becomes better than the previous generation.

The agent remembers its history using screenshots of the previous generations and chooses the best possible and most rewarding actions in the next generations ,thus improving .

WORKING :

- Without touching the underlying dynamics of the T-Rex game, the state-space would be a set of screenshots in the form of 1200x300 grid of RGB pixels. However, modeling this large state-space directly is difficult and computationally expensive. Thus, we apply preprocessing to raw pixels for data filtering and feature extraction. Different preprocessing procedure is adopted in different algorithms.

- Firstly we convert the image to grayscale, and then resize the image to 80 × 80 grid of pixels. Finally we stack the last 4 frames to produce an 80x80x4 input array.

- We used OpenCV , an open source computer vision tool for pixel-based feature extraction. Those features come from an intuitive understanding of how a human agent would play the game: identify the dinosaur, obstacles, their relative positions and velocities, and then decide the next action for T-Rex.

The steps for extraction of information from the screenshots:

1. **Background Filtering**
   The first step in extracting pixel-based features from the screenshots is to filter out useless pixels in prediction, such as the horizon and clouds. We also convert the image into grayscale to reduce the state-space.

2. **Object Detection :**
   After filtering the background, the objects are easily separated. We leverage OpenCV to detect the contours of the objects and find the corresponding bounding boxes. Each bounding box is represented by the x, y positions of the upper-left corner as well as the width and height.

3. **Object Classification :**
   Given bounding boxes, it is important to determine the type of each object. Since there are only three kinds of sprites (T-Rex, cactus, bird), we can use the width and height of the bounding boxes to classify the objects.

4. **Object Tracking :**
   The final step of pixel-based feature extraction is to calculate the speed of the obstacles and whether T-Rex is jumping. We track objects from frame to frame in order to measure the derivatives of each object on the screen. The tracking algorithm is a simple greedy matching of the nearest bounding box with the same type. By comparing the position of the detected object in two adjacent frames, we calculate the moving speed for each obstacle. Furthermore, by calculating the vertical speed of the T-Rex, we can infer whether the T-Rex is jumping up or dropping down.

- The optimization can be done using the backward propagation training method of CNN.

- If we train our CNN at every single frame, the training may suffer from several problems. Since the state at time t is highly correlated with state at time t+1, gradient descent after consecutive steps will cause erratic updates making the training very slow. In order to fix this problem and make the training faster, we use the **batch training method.**

REINFORCEMENT LEARNING (SUMMING UP) :

- The model extracts data from the screen captures and ties them up with the input that was given at that instance.

- If that instance turned out to increase the overall score of the game, it was rewarded and the reward value of the model was increased and it was included in the CNN model which was the basic model used for the Machine Learning.

- If the instance turned out to lead to the death of the dinosaur, it was punished and the overall reward value of the model was decreased.

- Now the RL Model will try to maximize its reward, so it will only continue with the approaches that led to reward and not punishment.

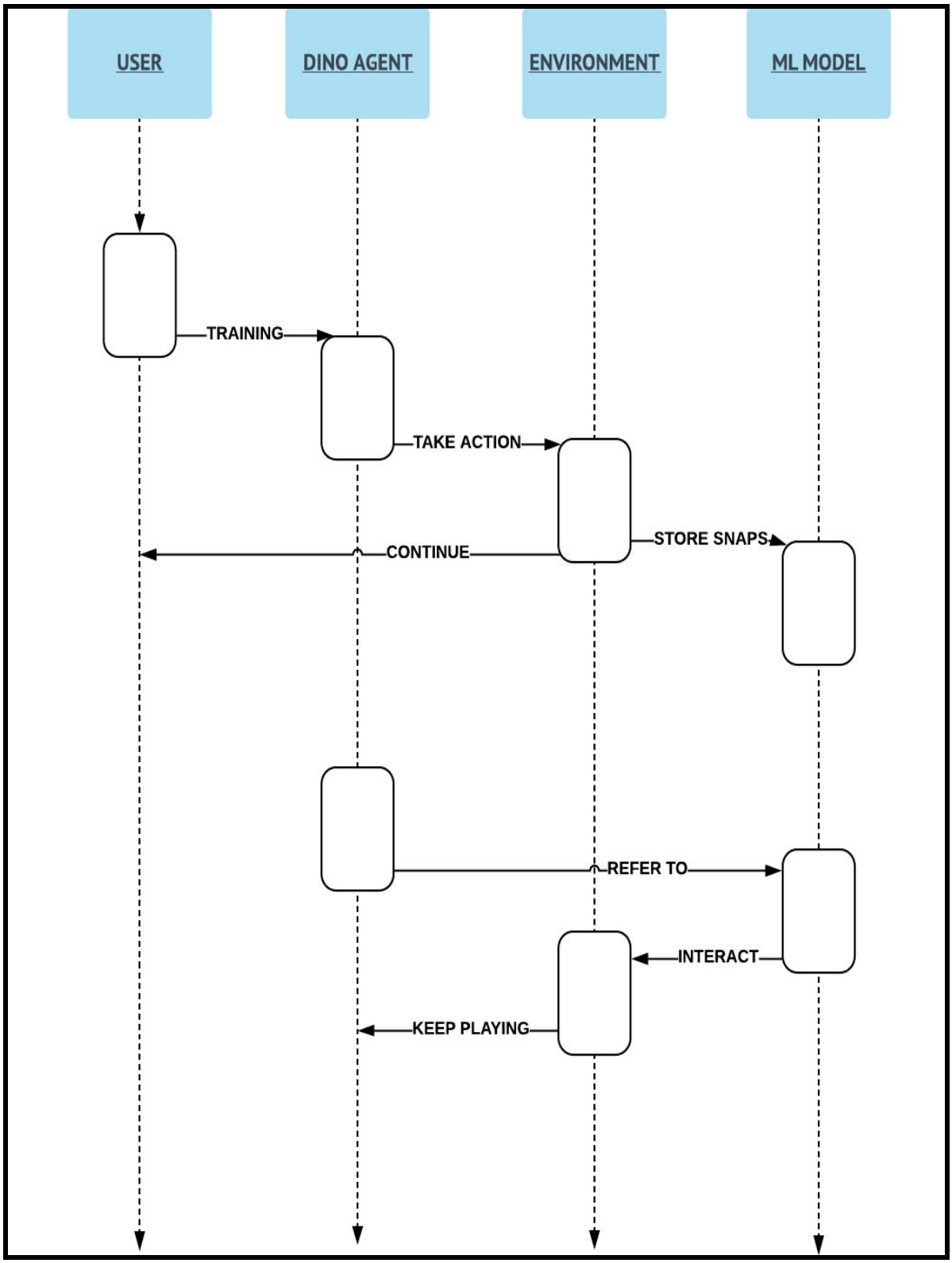
- After training the model with the above approach, it could with each run, go farther and farther in the game resulting in higher and higher reward.
- After consistently scoring above a human expert, the training is stopped .
- The model that was the most rewarded is stored for further use.

Libraries Used:

- Selenium
- Opencv
- Python-Imaging
- MSS
- Numpy

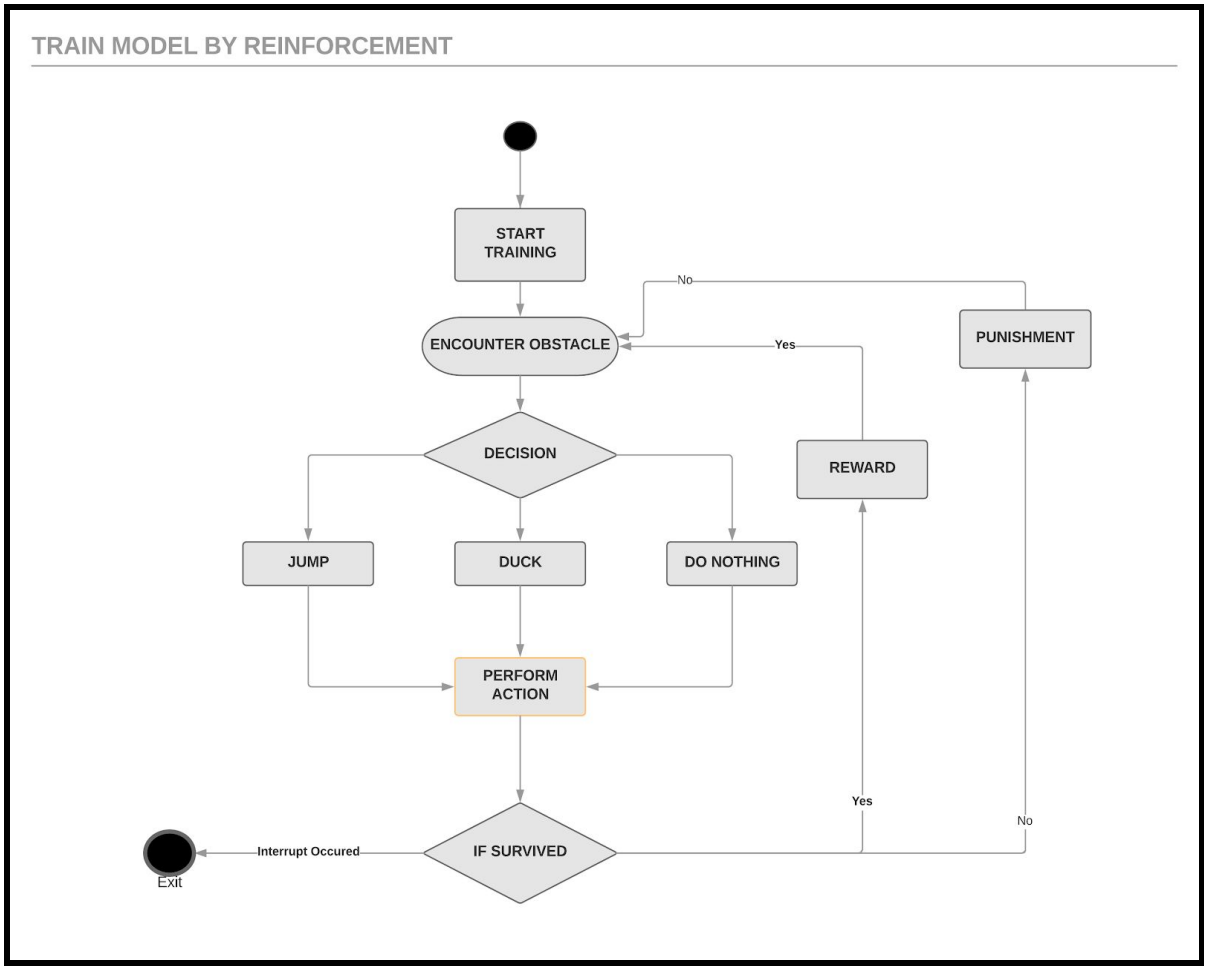
UML Diagrams:

Sequence Diagram:



ACTIVITY DIAGRAM:

1st part: Training the gamebot:



2nd part : Dino game played by the model itself: