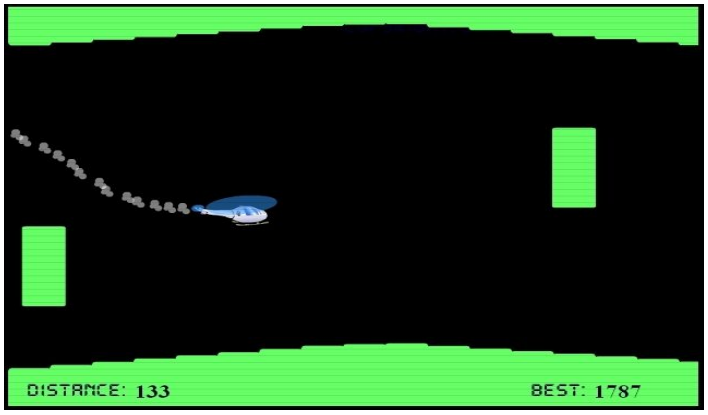
CSCI 5622 Final Project Proposal
Copter Bot based on Deep Q-Learning

Team Members:
Harivignesh Rajaram
Pranav Kumar SivaKumar
Saikrishna Jaliparthy

Our project:

Our objective is to develop a Deep Neural network model to automate the gameplay in Copter game. The Copter is a classic side-scrolling game where the agent must successfully navigate through a cavern.



Reasons to choose this Project idea:

Game bots are a popular means of exploring the application of Reinforcement learning. The paradigm of learning by trial-and-error, solely from rewards or punishments, is known as reinforcement learning (RL). With the inclusion of Deep learning concepts, the performance of automated game playing programs has seen much advancement that can go beyond the ability of even expert level human players.
Copter is an interesting classic game to work on with RL since it doesn't have a final goal state to achieve rather just the high score from staying alive for a longer time. It also provides us with the opportunity to upskill our Machine Learning knowledge and experience.

Why do we need Machine Learning ?:

Game bots for this can either be build using supervised learning or by reinforcement learning. But, If we want to build a bot using supervised learning, we need to have hundreds of hours of gameplay videos of world's best players playing this game which is tagged with their corresponding move at a particular frame which can then be given as input for the model. Given that we have such a dataset and high powerful GPU machines this problem can be tackled as a supervised classification problem. But instead, we can easily train a program by directly interacting with the game environment using a trial and error approach. Therefore Reinforcement learning would be the best choice for this kind of problems. The main difference in labels among the above techniques is that in supervised learning We have a target label for each of the training example and in reinforcement learning, We would have sparse and time-delayed labels – the rewards. Based only on those rewards the agent (the Copter) has to learn to behave in the environment.

Our Approach:

The most common method to formalize a learning problem is to represent it as a Markov decision process(MDP). A reinforcement learning algorithm called Q-Learning is utilized as it has been proven that for any finite MDP, at a given state Q-learning can find an optimal policy such that the reward is maximum by taking into account the successive states. For our problem, the state is the current position of the copter and the obstacles. The action is either pressing the up key or not. We plan to reward the agent for each successful second and punish it when it collides with the border or with an obstacle. With every game played, the copter observes the states it has been in and the actions it took. With regards to their outcomes, it punishes or rewards the state-action pairs.

Our goal is to progress from a simple Multi-layered perceptron (MLP) to experimenting with a convolutional neural network with a couple of fully connected layers and more complex additions. The input to the MLP involves all features like velocity, a distance of the player to the ceiling/floor, to the block’s top/bottom, number of lives and the current action. Similarly to the concept where we intend to use CNN's, the input data are the pre-processed frames of the rendered game. The output from either of the networks are the Q-values of each possible actions, they are continuous values that make it a regression task that can be optimized using a loss function.

We also plan to implement Experience Replay which will allow our network to train itself using stored memories from it’s experience. When the time comes to train, we simply draw a uniform batch of random memories from the buffer, and train our network with them. This may prevent overfitting the agent to the recent events.

Dataset:

As we will be implementing reinforcement learning technique to train a bot, we will not be using any external datasets, Instead, training datasets were collected by interacting with the OpenAI Gym Interface / PyGame Learning environment.