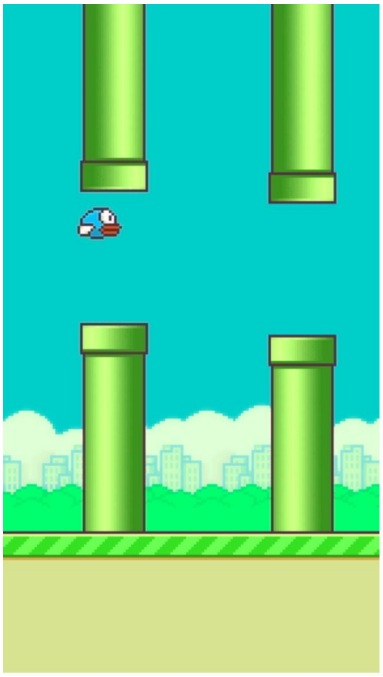
Capstone Proposal

Domain Background

The Capstone Project is dedicated to solving a Reinforcement Learning problem.
RL - is booming branch of Machine Learning, which could solve plenty of actual problems of our time and have a significant impact on such industries as robotics, finance, gaming and even chemistry! Most of RL algorithms are applicable for multiple tasks (e. g. in this paper PPO algorithm was used both to train humanoid to run and to train the agent to play different arcade games).
That is why I have a deep interest in RL and would like to solve one of the RL problems to get a better understanding of different existing RL algorithms and be able to solve more challenging problems in future.

Problem Statement



The main goal of the project is to teach the agent to play Flappy Bird game. Flappy Bird is a side-scrolling game where the agent must successfully navigate through gaps between pipes.

Datasets and Inputs

As it is the RL problem, the primary source of information here is the environment.
In the project, I will use PyGame Flappybird environment to develop the agent that can play this game a certain amount of time without failing. The environment is initialized with parameters:

- `width(default=288)`
- `height(default=512)`
- `pipe_gap(default=100)`.

In addition to the visual representation (game screen), the environment state is also described by a dictionary with the following values:

- player y position
- player velocity
- next pipe distance to player
- next pipe top y position
- next pipe bottom y position
- next next pipe distance to player
- next next pipe top y position
- next next pipe bottom y position

The agent could take one of two possible actions: *0 - do nothing, 1 - go up.*
For each pipe it passes through it gains a positive reward of +1. Each time a terminal state is reached, it receives a negative reward of -1. The game ends if the player makes contact with the ground, pipes or goes above the top of the screen. So the goal is to keep playing as long as possible, avoiding obstacles.

Benchmark Model

As a benchmark model I could use one of the top models on the OpenAI Gym Leaderboard. These models are complicated and have fantastic scores, so these are good examples of how the agent should perform. The highest score on the Leaderboard is 264 and the algorithm that achieved such result is PPO. The second highest score of 261 has the agent trained using AC3 algoritm.
However, there also some papers (Game Playing with Deep Q-Learning using OpenAI Gym, Playing Flappy Bird with Deep Reinforcement Learning) where the main algorithm is Deep Q-network and the "good" **average** scores are around 40 - 60 points.
So my goal is to get the best 100 episodes average score of at least 40 points.

Solution Statement

As it was told above, the most popular algorithm for solving this problem is Deep Q-network. As this discrete action space, Deep Q-network seems like a good starting point. The highest Leaderboard score has an agent trained using the PPO algorithm. As this is an educational project, in my opinion, it is also a good idea to implement PPO and compare results with DQN.

Evaluation Metrics

Another advantage of this Reinforcement Learning problem is that it's easy to evaluate model performance using the game score.
The highest score on the Leaderboard is 264.0. I'm not sure that my agent would beat this result, but it would be nice to have the score at least as high as one of the Leaderboard models.

Project Design

The first steps would be to research techniques described in papers and try to implement them to reproduce the results. Next, try to tune hyperparameters and add some improvements to the existing algorithm. Finally, try to use the PPO algorithm to solve this problem and compare the results.

The reasonable basic preprocessing of the game image would be:

- converting to grayscale
- noise reduction
- scaling to a smaller image
- normalization

The starting DQN architecture would consist of 3 convolution layers and two fully connected layers (exact sizes are still not precise, but they shouldn't be too large). Also always a good idea to play with Batch normalization, different regularization techniques, and optimizers. Another problem to solve - finding a balance between exploration and exploitation. Prioritized experience replay also could boost learning.