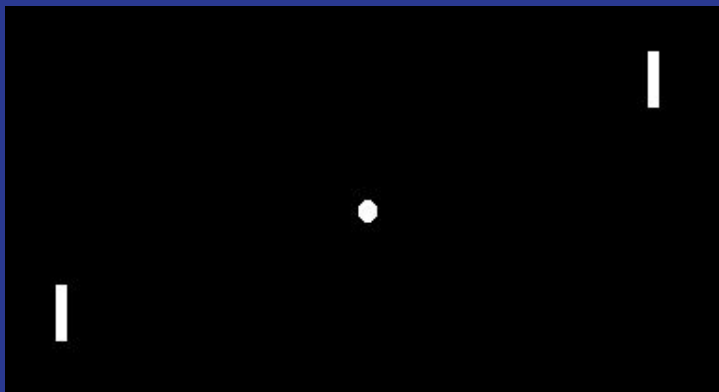


# Battle of Pong Agents: Deep Q-Learning vs. Policy Gradient

Yash Bhutwala, Matt McNally, Kenny Rader, John Simmons



# Problem

Given the following:

- A sequence of images (frames) representing each frame of the Pong game.
- An indication when we've won or lost the game.
- An opponent agent that is the traditional Pong computer player.
- An agent we control that we can tell to move up or down at each frame.

Can we use these pieces to train our agent to beat the computer? Moreover, can we make our solution generic enough so it can be reused to win in games that aren't pong?



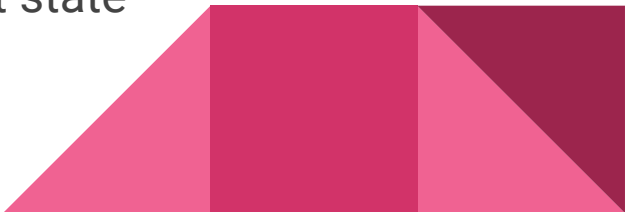
# Goal of the Project

- Existing Solutions:
  - Deep Q-Learning (DQN): “[Human-level control through deep reinforcement learning](#)”. 2015.
    - DeepMind, AlphaGo
  - Policy Gradient (PG): “[Asynchronous Methods for Deep Reinforcement Learning](#)”. 2016.
- The goal of our project is to create two agents, one using DQN and one using policy gradients, that are both trained in the same environment, OpenAI Gym’s Pong, and compare the performance of the two algorithms.



# DQN Approach

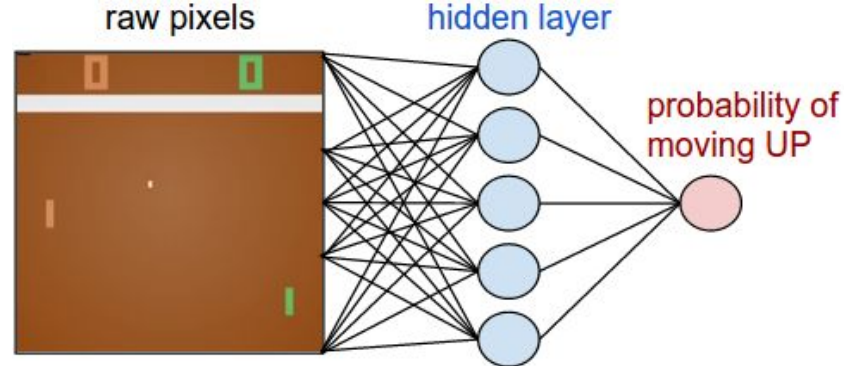
## Basic DQN Algorithm

- Initialize a replay memory and a Q function as a convolutional neural network
  - Choose the action with the highest Q value, calculated using our current state, otherwise choose a random action based on some probability function
  - Observe reward and store state, action, reward, and new state in replay memory. Keep a running score of all the rewards for all episodes.
  - Propagate error back through the network to train the Q function
  - If new state is terminal state then stop, otherwise set state to new state.
- 

# PG Approach

Use a 2-layer Neural Network to:

- Take in images from the game and preprocess them.
- Use the Neural Network to compute a probability of moving up.
- Sample from that probability distribution and tell the agent to move up/down.
- If the round is over, find whether you won or lost.
- When the episode has finished (someone got to 21 points), pass the result through the backpropagation algorithm to compute gradient for the weights.
- After 10 episodes have finished, sum up the gradient and move the weights in the direction of the gradient.
- Repeat this process until weights are tuned enough .

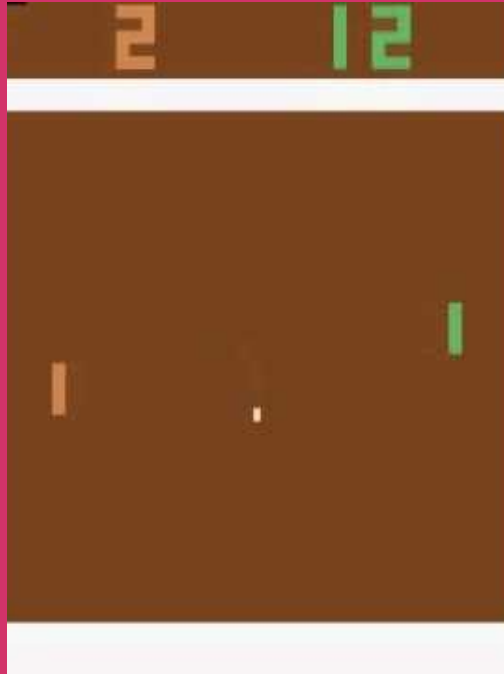


# Metrics

- Train each agent for 5000 episodes.
- Now run each agent for 100 episodes, and capture:
  - Episodes won
  - Running mean of the rewards

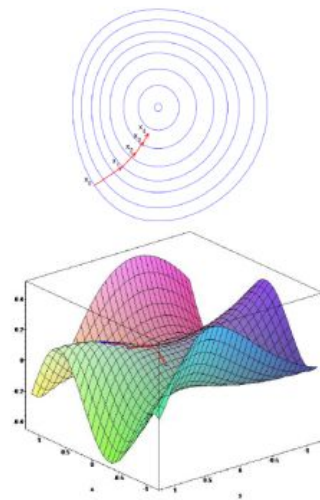


# DEMO



# PG Extra

- In ordinary supervised learning we would have access to a label, such that we know what the correct thing to do is.
- However, what do we do if we do not have the correct label in the Reinforcement Learning setting?
- We have a stochastic policy that samples actions and then actions that happen to eventually lead to good outcomes get encouraged in the future, and actions taken that lead to bad outcomes get discouraged.
- ***Policy Gradients: Run a policy for a while. See what actions led to high rewards. Increase their probability.***





# Supervised vs. PG

