TRAINING PONG TO BE UNBEATABLE

Machine Learning - Fall 2024
Sam Wisnoski

My Goal:

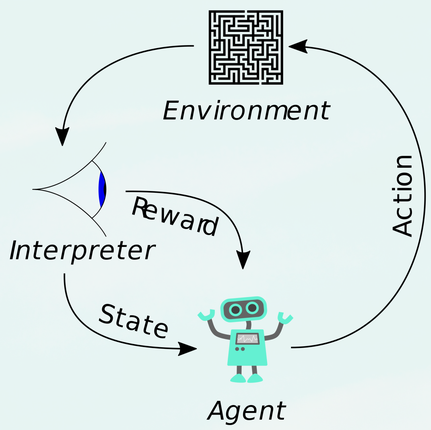
Use Open AI Gymnasium's Pong Environment and Reinforcement Learning to train an algorithm to be (basically) unbeatable at Pong.

Learning Goals!!

- Learn RL theory and implementation using RLlib and custom code.
- Apply RL to control video game agents.
- Explore RL in automated robotics.
- Train agents in OpenAI Gym across diverse environments.

What is Reinforcement Learning?

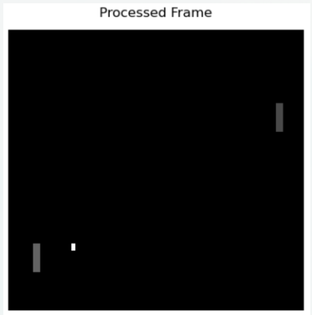
Reinforcement Learning is a machine learning approach where an agent learns by interacting with its environment (it's state) and receiving feedback in the form of rewards or penalties. The agent balances exploration and exploitation to optimize behavior, making RL ideal for dynamic problems like video games or robotics.



How do we apply RL to Pong?

Step One: Process the Gamestate

To prepare the Pong game frames for our reinforcement learning model, we need to reduce their complexity whilst retaining critical gameplay information. To do this, we can evaluate each pixel in the frame individually and turn it into a numpy array using the "resize_frame" function. We resize the frame to 84x84 pixels, removing the scoreboard, and then grayscale the frame. This ensures that the input to our model is standardized and compact.



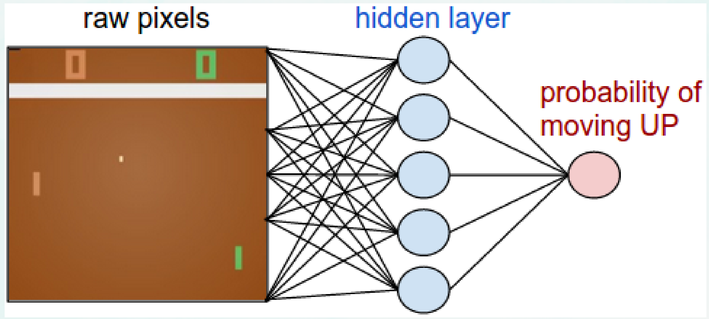
Step Two: Create a "Memory" of Experiences

The agent stores "experiences" in memory, each containing. Experience Replay improves training by storing past experiences in a memory buffer and sampling random batches instead of consecutive ones. This breaks correlations, reinforces rare events, improves data efficiency, and stabilizes learning by reducing fluctuations in Q-value updates.

Step Three: Build our Model:

The Deep Q-Network (DQN) maps states (sequence of frames) to Q-values for each action:

- Input: 84x84 grayscale frames, 4 stacked for motion context.
- Convolutions: Three layers extract spatial and temporal features, refined through ReLU activation.
- Dense Layers: Flattened features processed by a dense layer (512 neurons), with outputs providing Q-values.
- Loss & Optimizer: Uses Huber Loss for robustness and Adam Optimizer for efficiency.



A fundamental part of this model is the relationship between exploration and exploitation. Exploration involves taking random actions to discover new strategies, while exploitation chooses actions with the highest predicted rewards. Over time, the agent shifts from exploration to exploitation, ensuring it learns effectively while optimizing decisions.

What is the Math Behind This?

The Bellman equation is key to reinforcement learning, calculating Q-values that represent the total expected reward for taking an action in a given state and following the optimal policy thereafter.

$$Q(S_t, A_t) = (1 - \alpha)\, Q(S_t, A_t) + \alpha * (R_t + \lambda * max_a\, Q(S_{t+1}, a))$$

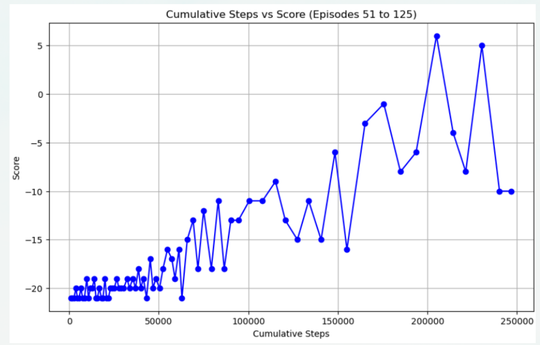
It updates Q-values by combining the immediate reward with future rewards, balancing short-term and long-term planning. In a Deep Q-Network (DQN), this equation helps the agent refine its predictions and make better decisions over time, ultimately maximizing overall rewards.

Importance in the Non-Digital World

Reinforcement learning mirrors real-world learning through trial, error, and feedback, offering powerful solutions in areas like renewable energy, healthcare, and robotics.

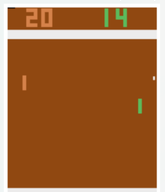
However, it raises ethical concerns, such as biases in reward systems, the "black-box" nature of models, and unintended behaviors in high-stakes applications like self-driving cars or military use. Do we trust companies to prioritize rewarding safety or humanitarianism over profit? As Richard Feynman said, "Every man is given the key to the gates of heaven. The same key opens the gates of hell. And so it is with science."

Results!



The stable_baselines3 model, after 36 hours of training, performed the best overall, with scores ranging from -4 to 15, though it started to stagnate toward the end of training. My homebuilt model, trained for 28 hours, achieved scores between -19 and -11.

Custom Model (Lost)



Pre-Built Model (Won)