Human level performance of an AI playing Donkey Kong

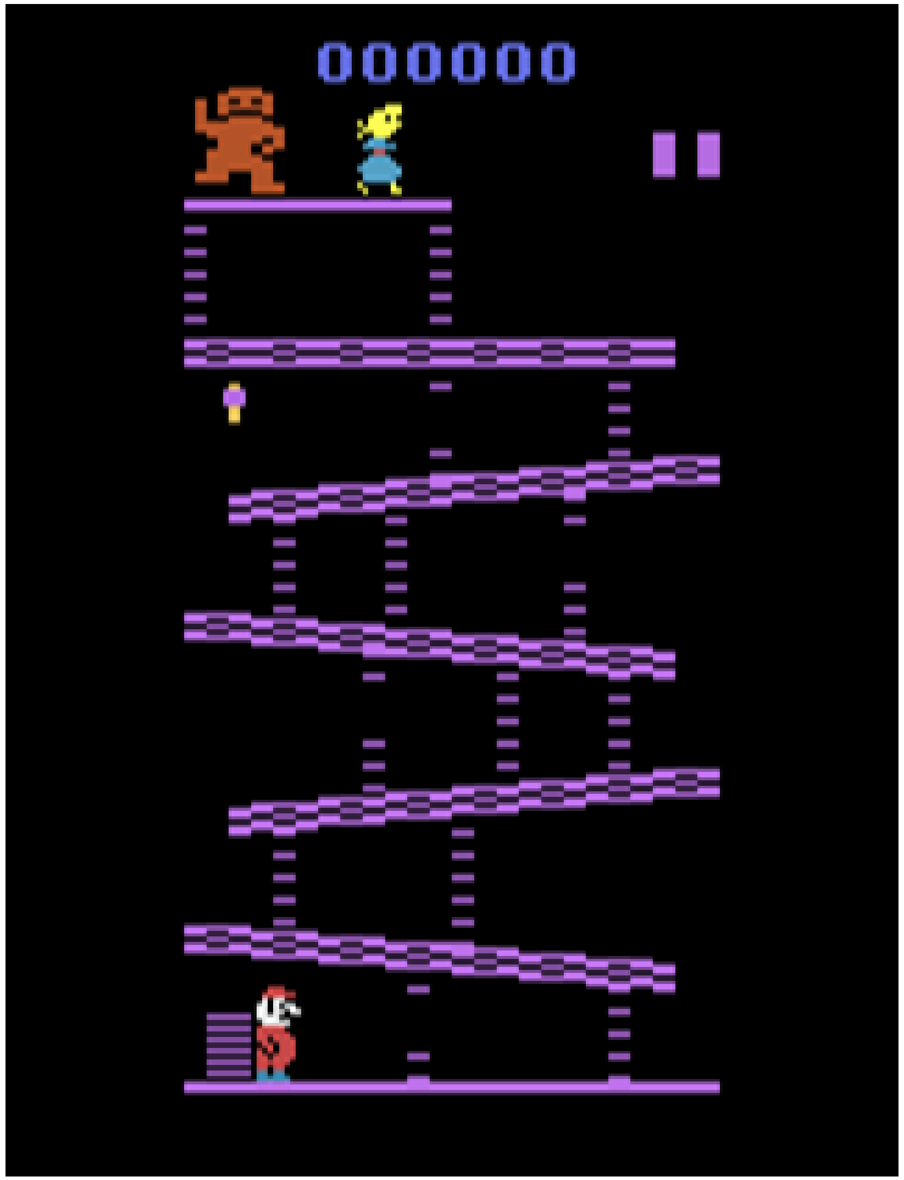
Jakub Grzywaczewski, Anna Ostrowska, Igor Rudolf, Marta Szuwarska

[1]

Contents

Donkey Kong

- Donkey Kong is an arcade video game in which the player takes on the role of Mario and aims to rescue his girlfriend from a giant gorrilla called Donkey Kong.
- In order to get to his girlfriend, Mario needs to climb the unbroken l adders and omit the barrels.
- In addition, if Mario grabs the hammer, he can hit the barrels.



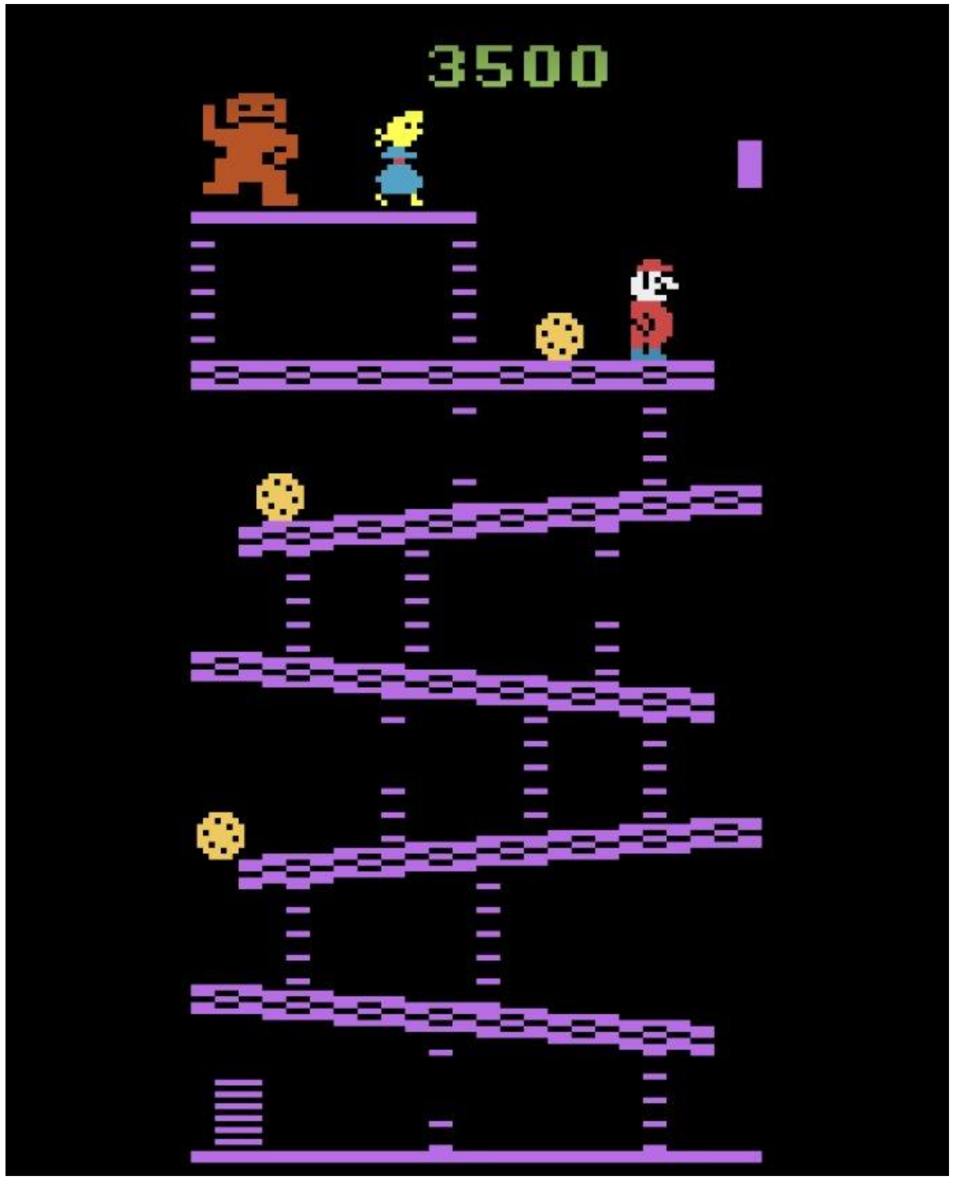
Objective

The primary objective:

Reach the first level successfully

Secondary objective:

Do it better, faster than a human



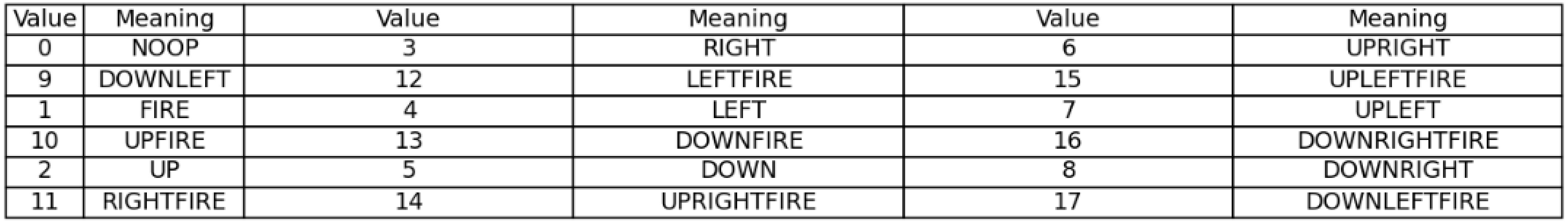
API

Gymnasium

An API standard for reinforcement learning with a diverse collection of reference environments

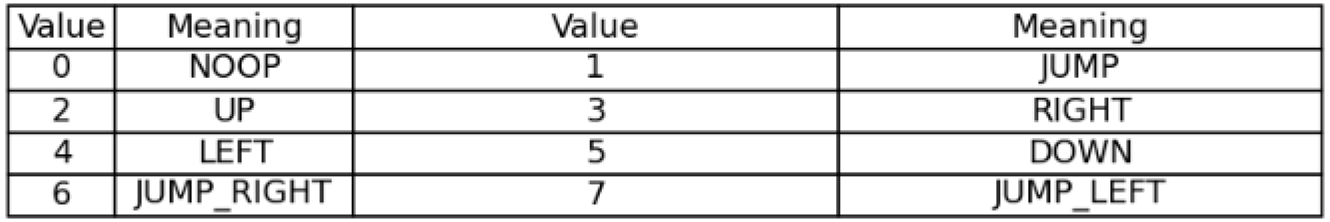
Set of Actions

Set of possible actions included in the Gymnasium and their meanings:



Value	Meaning	Value	Meaning	Value	Meaning
0	NOOP	3	RIGHT	6	UPRIGHT
9	DOWNLEFT	12	LEFTFIRE	15	UPLEFTFIRE
1	FIRE	4	LEFT	7	UPLEFT
10	UPFIRE	13	DOWNFIRE	16	DOWNRIGHTFIRE
2	UP	5	DOWN	8	DOWNRIGHT
11	RIGHTFIRE	14	UPRIGHTFIRE	17	DOWNLEFTFIRE

Set of possible actions chosen by us:



Value	Meaning	Value	Meaning
0	NOOP	1	JUMP
2	UP	3	RIGHT
4	LEFT	5	DOWN
6	JUMP_RIGHT	7	JUMP_LEFT

DQN

- **Deep Q-Networks (DQN):** algorithm that combines Q-learning, a model-free reinforcement learning algorithm, with deep neural networks to handle high-dimensional state spaces.

- Key concepts:
    - Q-learning
    - Deep Neural Networks
    - Experience Replay
    - Split Target and Value Networks

DQN explanation

- **Q-learning**: value based algorithm. It seeks to learn a policy that maximizes the total reward by learning the action-value function Q(s,a) , which represents the expected reward of taking action in a state s. The Q-function is updated using the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

- **Deep Q-Networks (DQN)**: uses a neural network to approximate the Q-function. Useful when we deal with large or continuous state spaces
- **Experience replay**: stores the agent's experiences (state, action, reward, next state) in a replay buffer and samples mini-batches of experiences to train the neural network.
- **Target Network**: is a copy of the Q-network that is used to generate the target values during training. The target network is updated less frequently than the Q-network to improve stability.

SARSA

- **Sarsa (State-Action-Reward-State-Action):** is an on-policy reinforcement learning algorithm used to solve Markov Decision Processes (MDPs).Unlike Q-learning, which is off-policy and uses the maximum future reward to update the Q-value, SARSA uses the action taken by the current policy to update the Q-value.

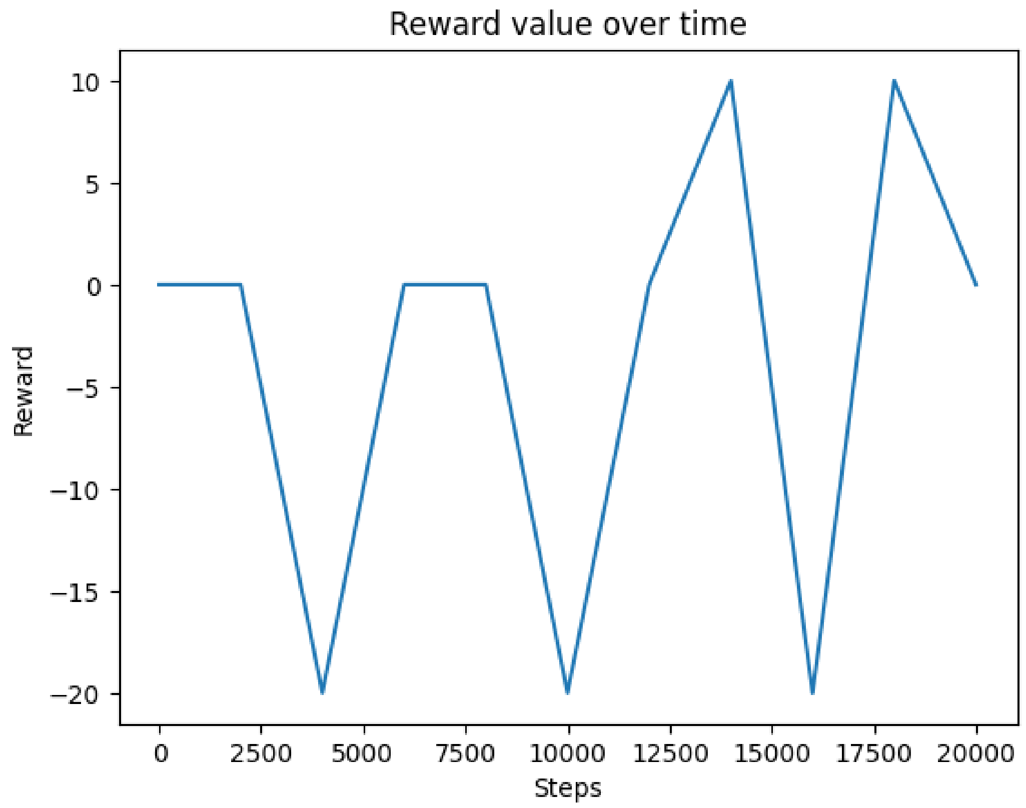
- Q-value update: $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$

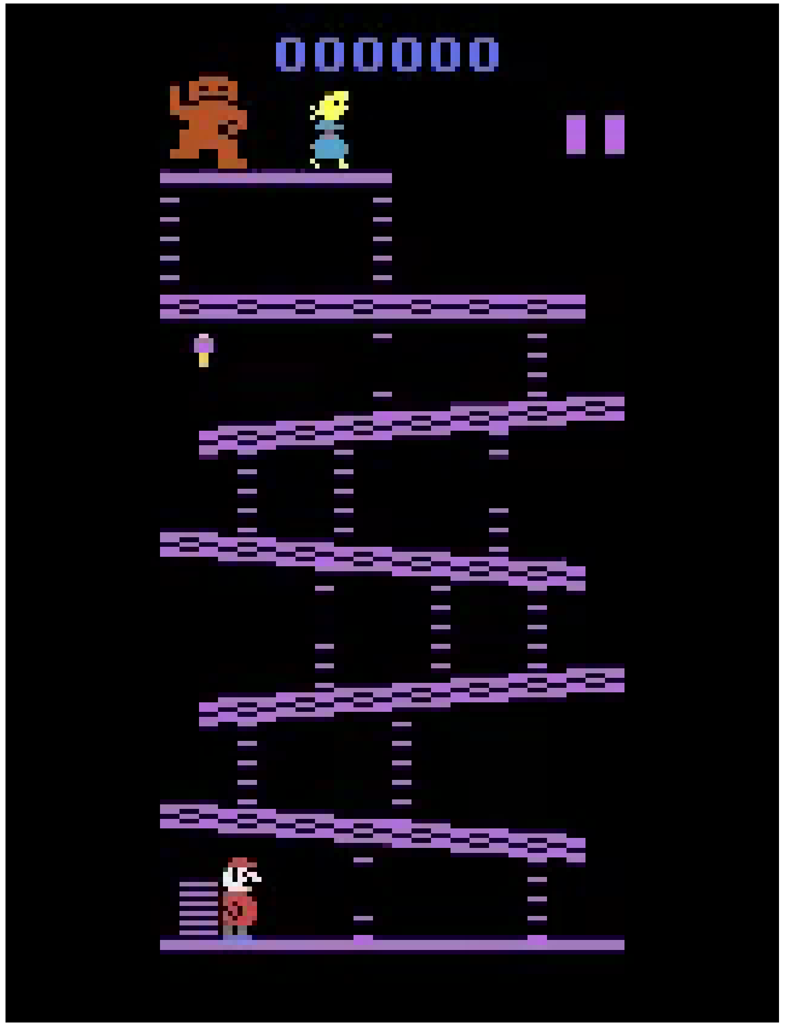
- Epsilon-greedy policy: With probability epsilon choose random action. With probability 1-epsilon, choose the action that maximizes the Q-value.

$$a = \underset{a'}{\arg\max}\, Q(s',a')$$

Baseline

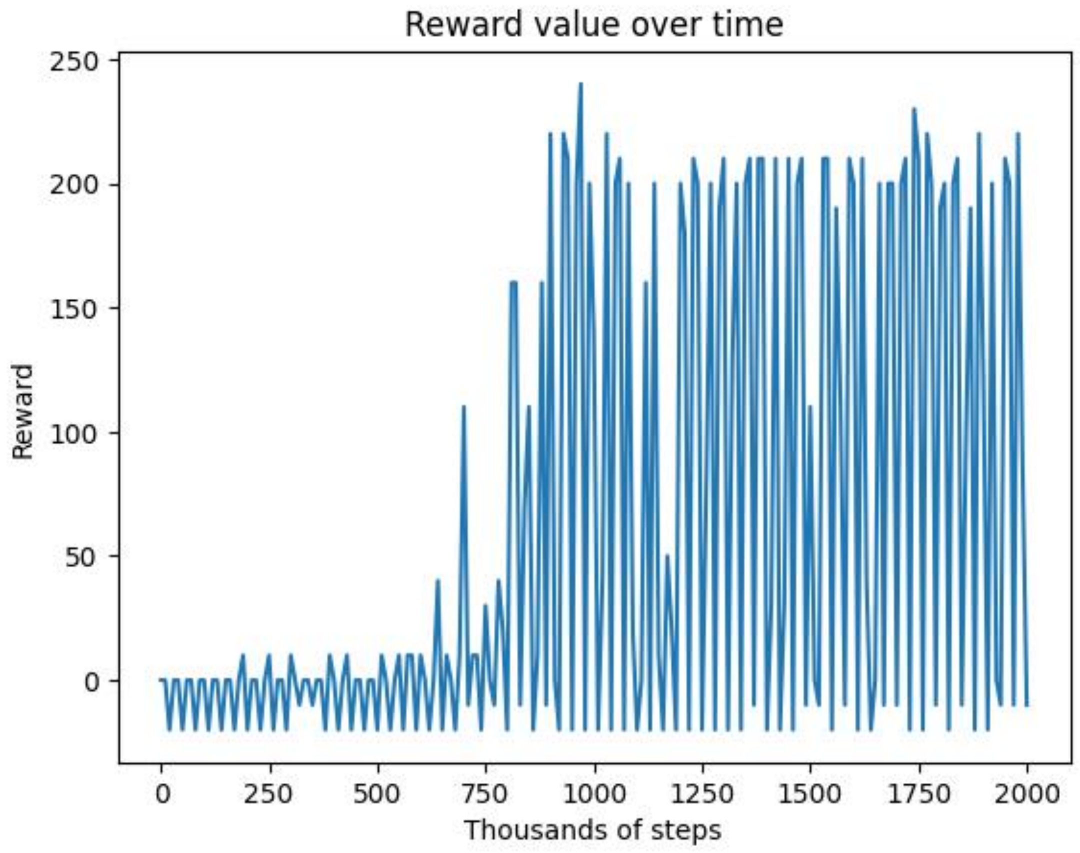
At first, we only punished the agent for dying (PD), which resulted in the agent staying at the first level moving away from the barrels as much as he can.

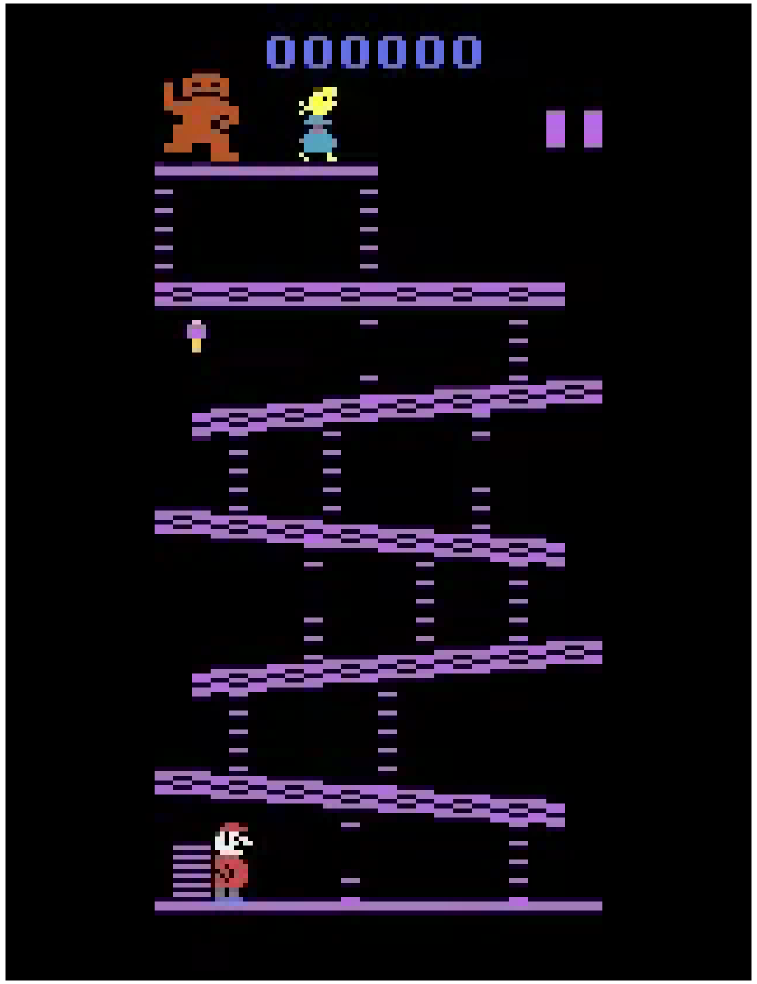




Baseline

With time the agent learned to jump over barrels.



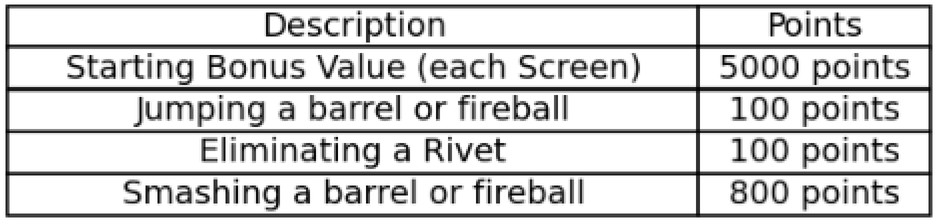


Baseline

Why does it not work?

While you control Mario, the Score/Bonus Indicator shows your bonus counting down. Between screens and at game end, your score is displayed.

Scoring:



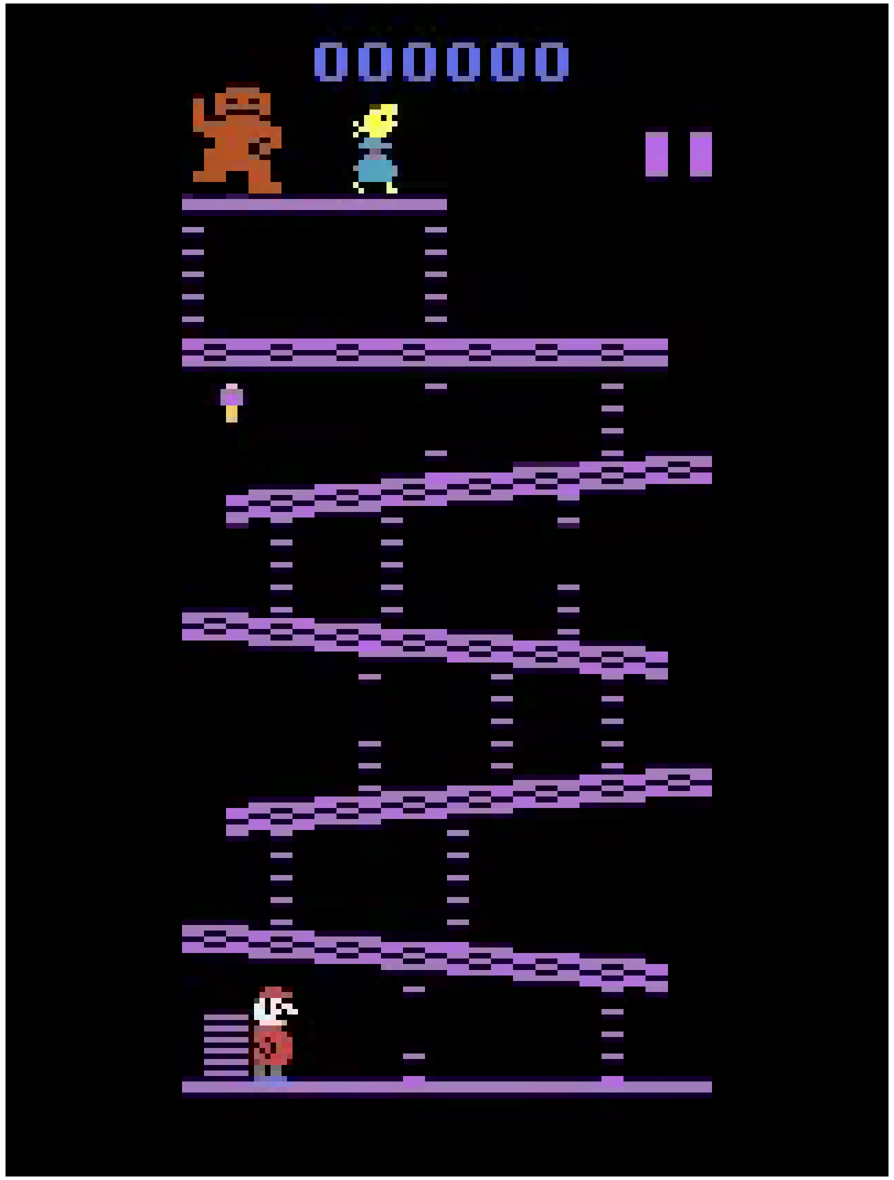
Description	Points
Starting Bonus Value (each Screen)	5000 points
Jumping a barrel or fireball	100 points
Eliminating a Rivet	100 points
Smashing a barrel or fireball	800 points

Ideas after consulting

- Severe punishment for needless jumping
- Rewarding going up and punishing going down while on a ladder
- Odds of specific actions dependent on the altitude level
- Evaluation without exploration every n episodes
- High reward for following magic stars path

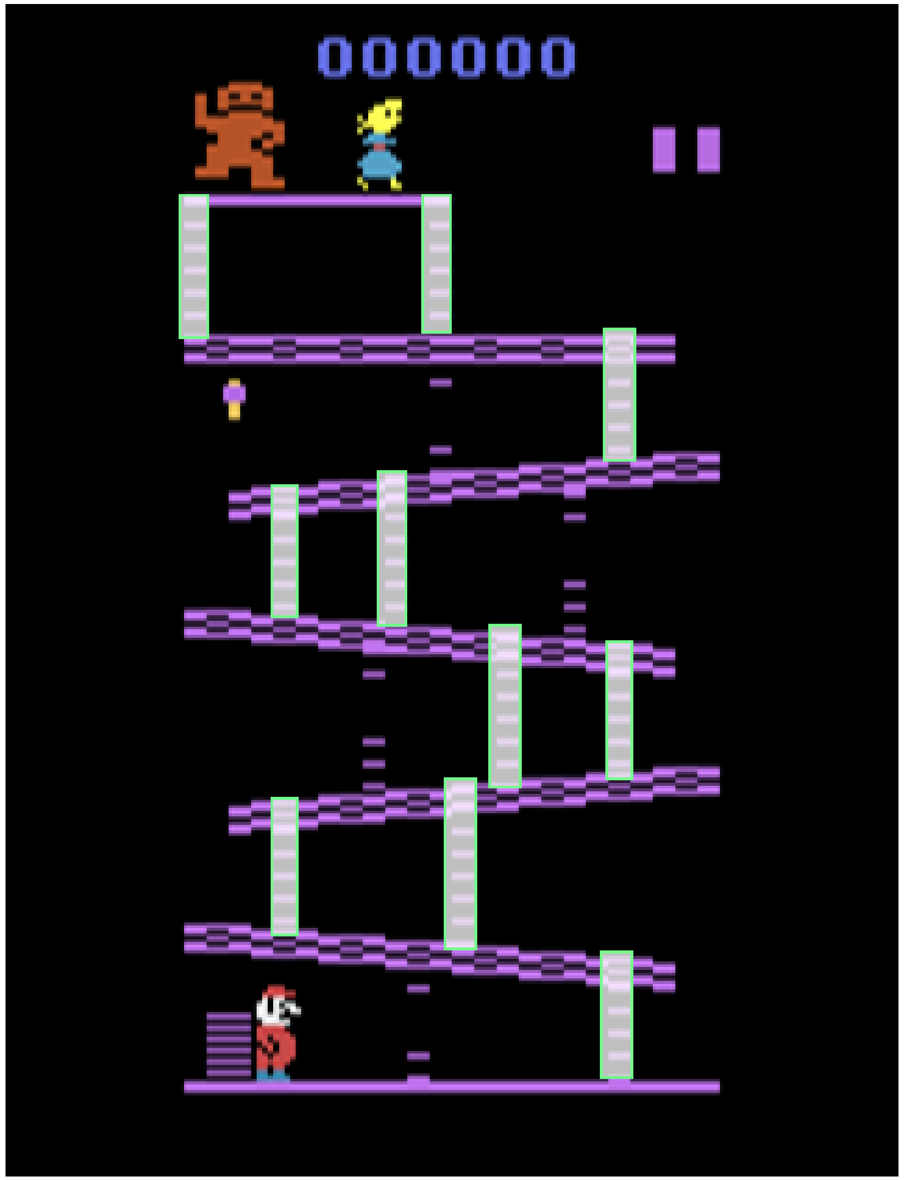
Punishing needless jumping (PNJ)

- We added a function that checks if there is any barrel near Mario.
- Every time Mario was not near the barrel or on the ladder and jumped up, we gave him a negative reward.



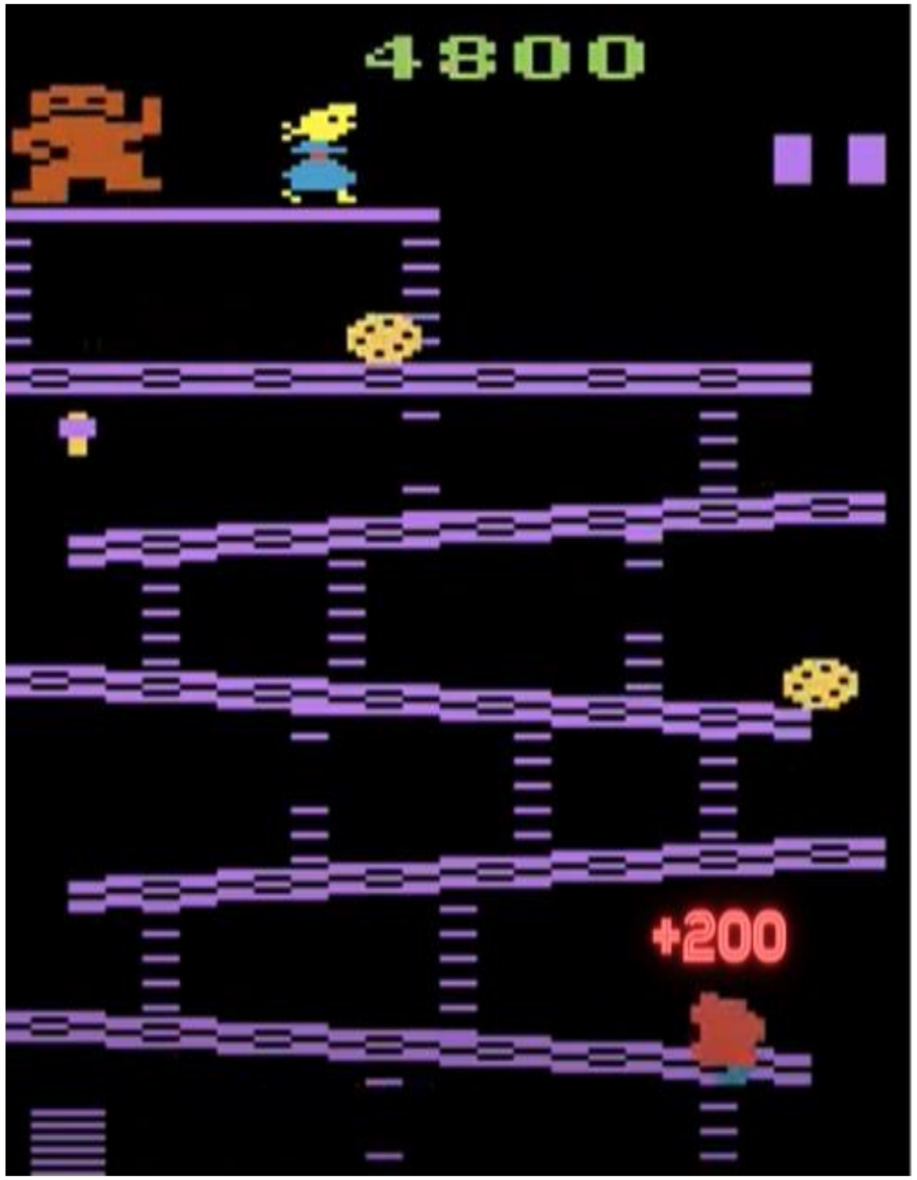
Ladders incentive (LAD)

- We found pixels corrensponding to ladders manually and then tested whether they actually coincide with the field where Mario can go up.
- Then we modified the punishment for needless jumping so that Mario is not punished if close to the ladder.
- We also added a reward for moving up on a ladder.



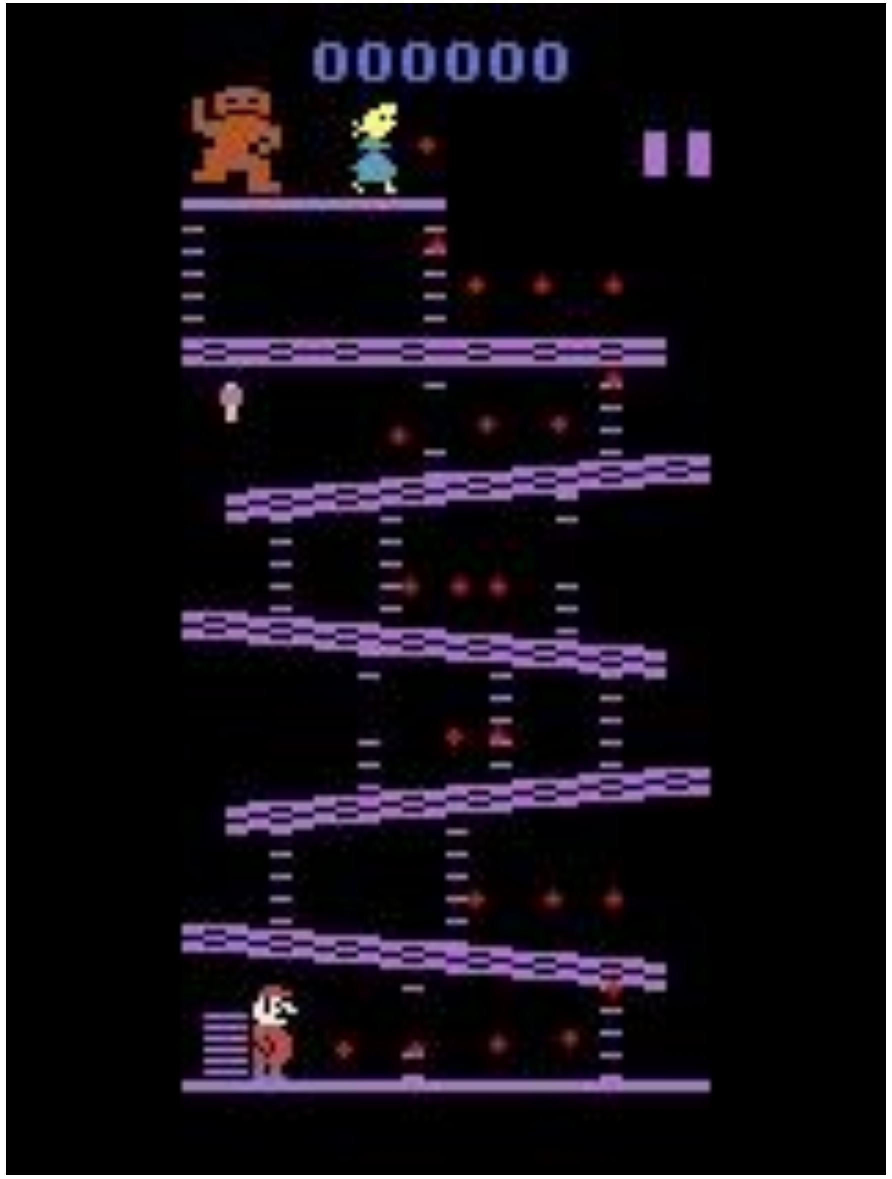
Level incentive (LVL)

- We added a reward for climbing to the next altitude level.



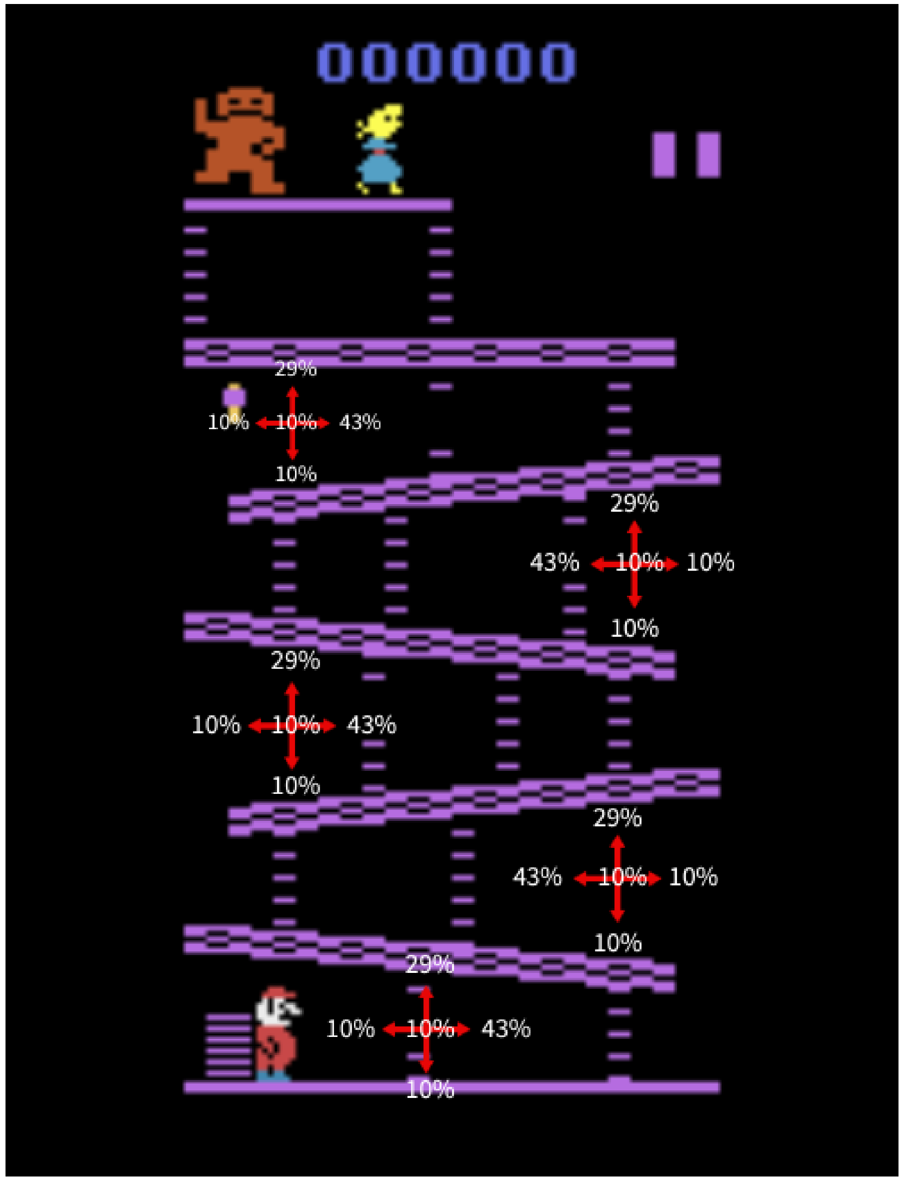
Magic stars incentive (MS)

- We chose the path Mario would have to take to get to the princess and placed some points ("stars") on it.
- The moment Mario reached the area of the star, we gave him a reward for it and removed the star from the available to get in a given game.



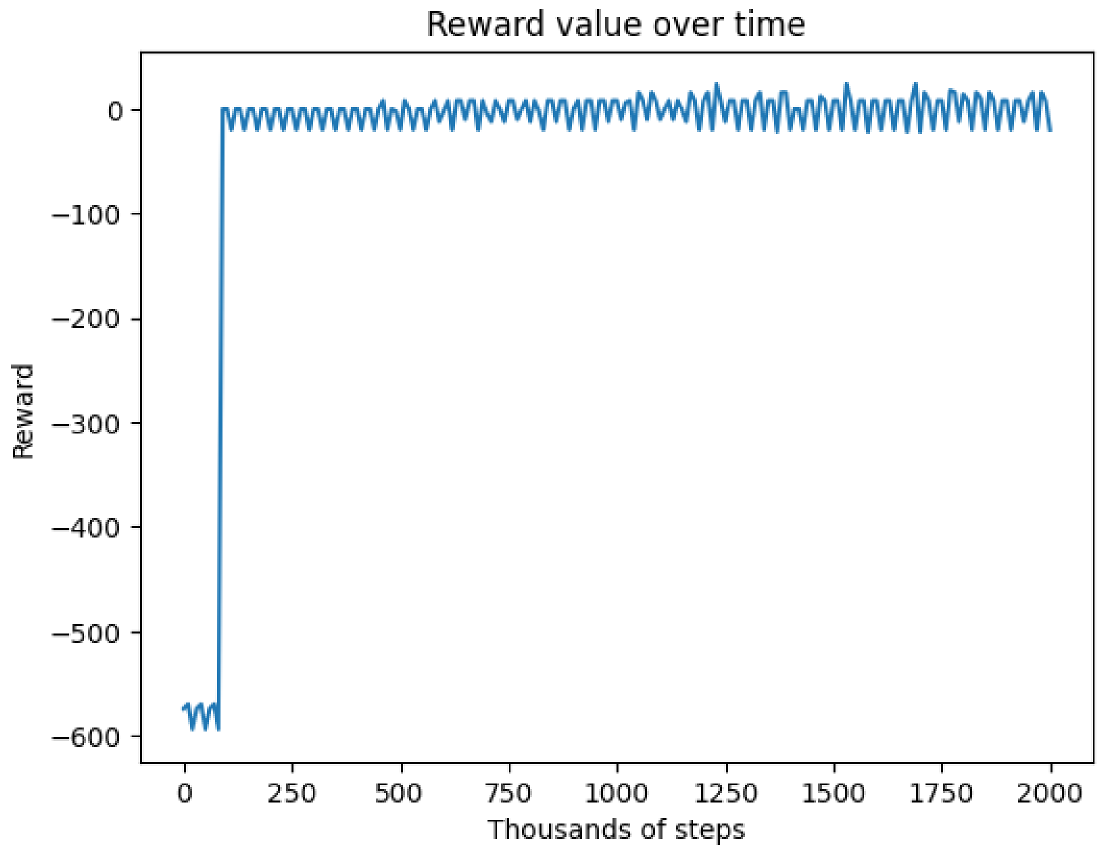
Action heuristics (AH)

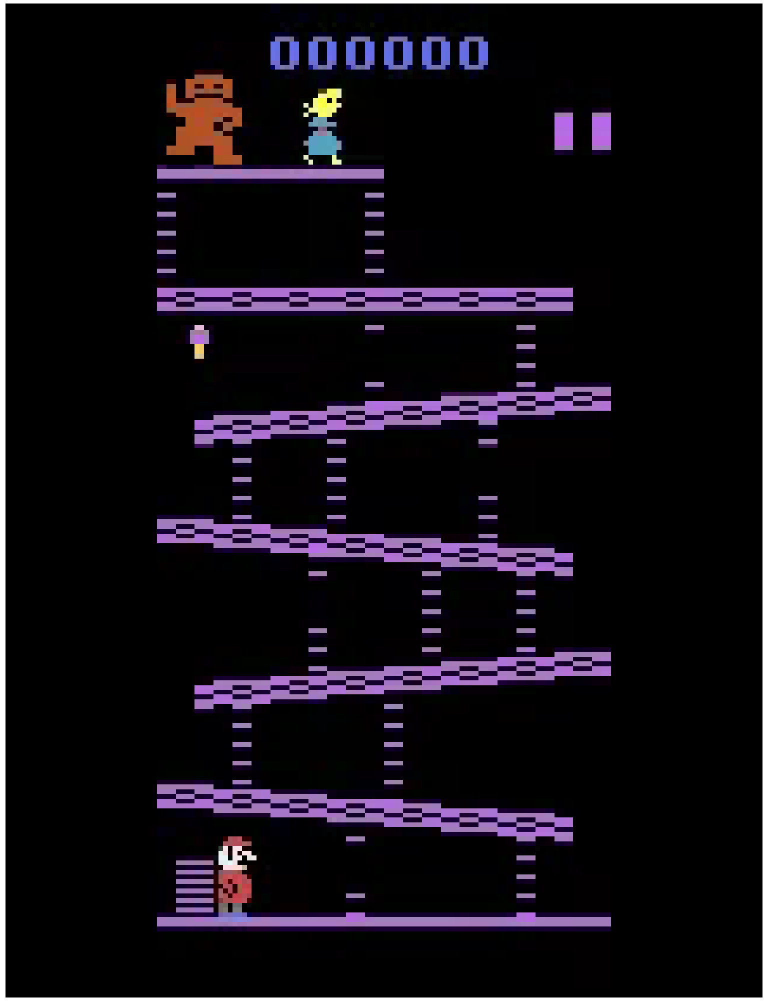
- We wanted to take control over actions performed by Mario. One way of doing this was to create discreate distribution depending on Mario's current level.



Punishing needless jumping

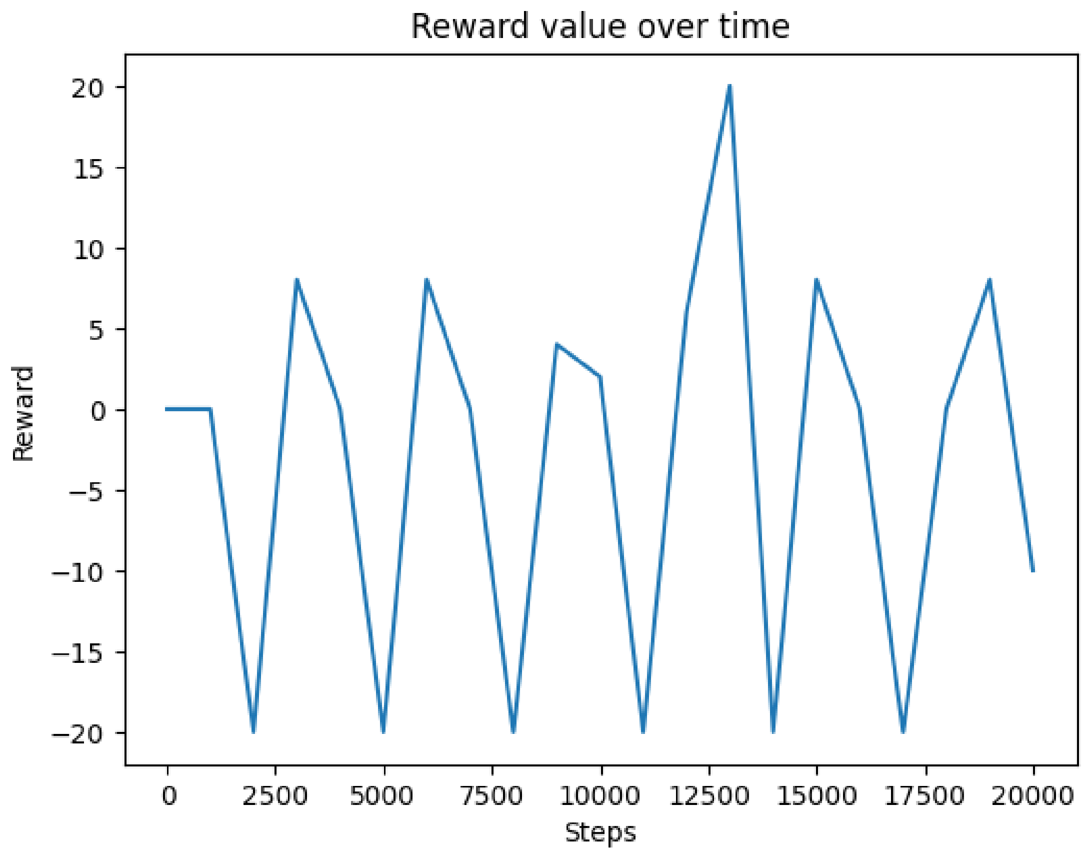
Adding punishing needless jumping only resulted in the agent going back and forth.

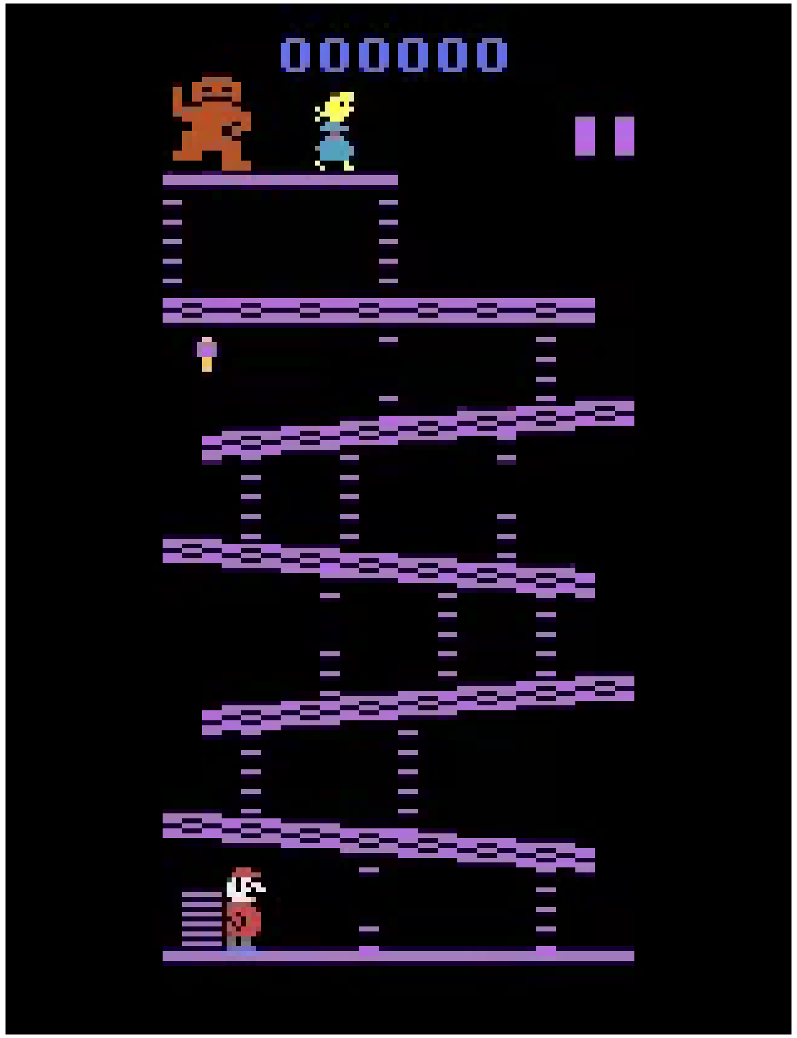




Magic stars incentive

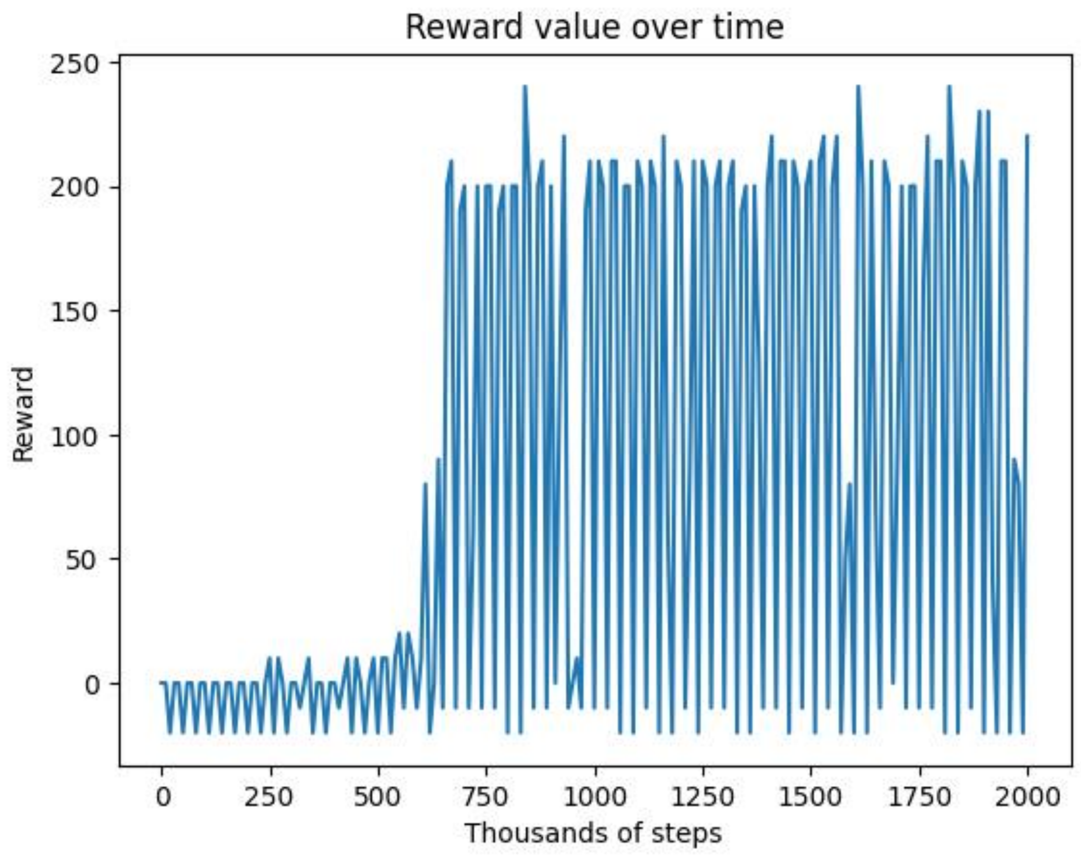
Adding magic stars incentive only resulted in the agent walking in circles.

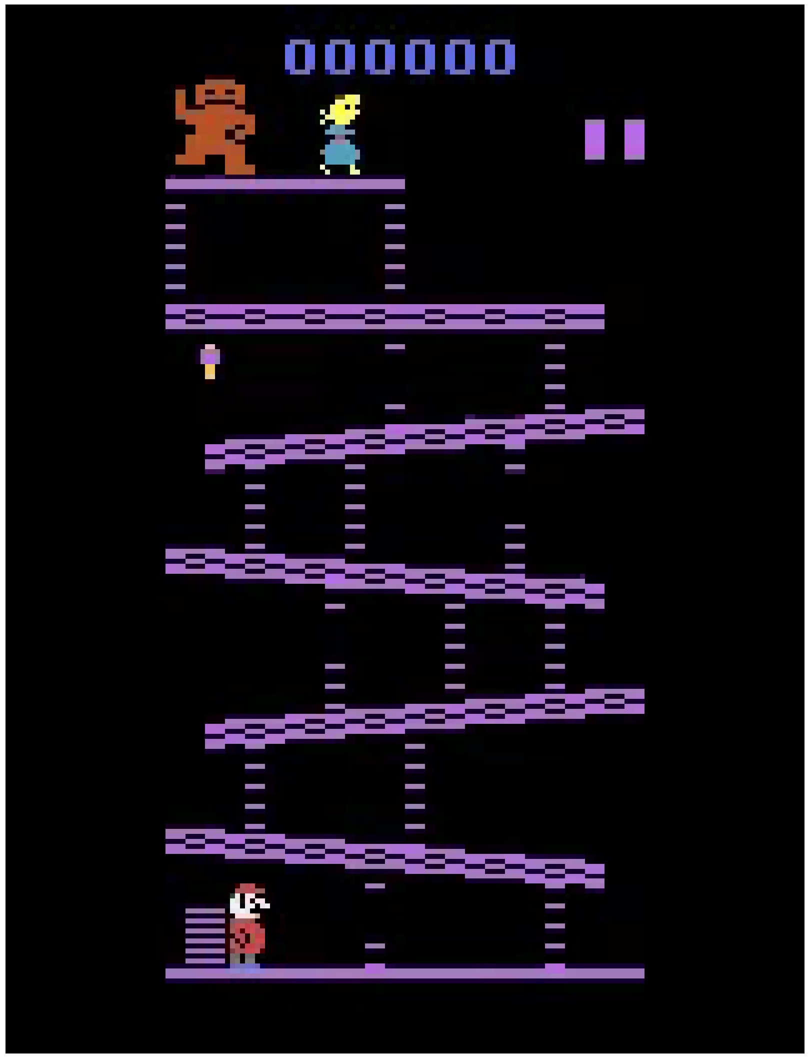




Level incentive

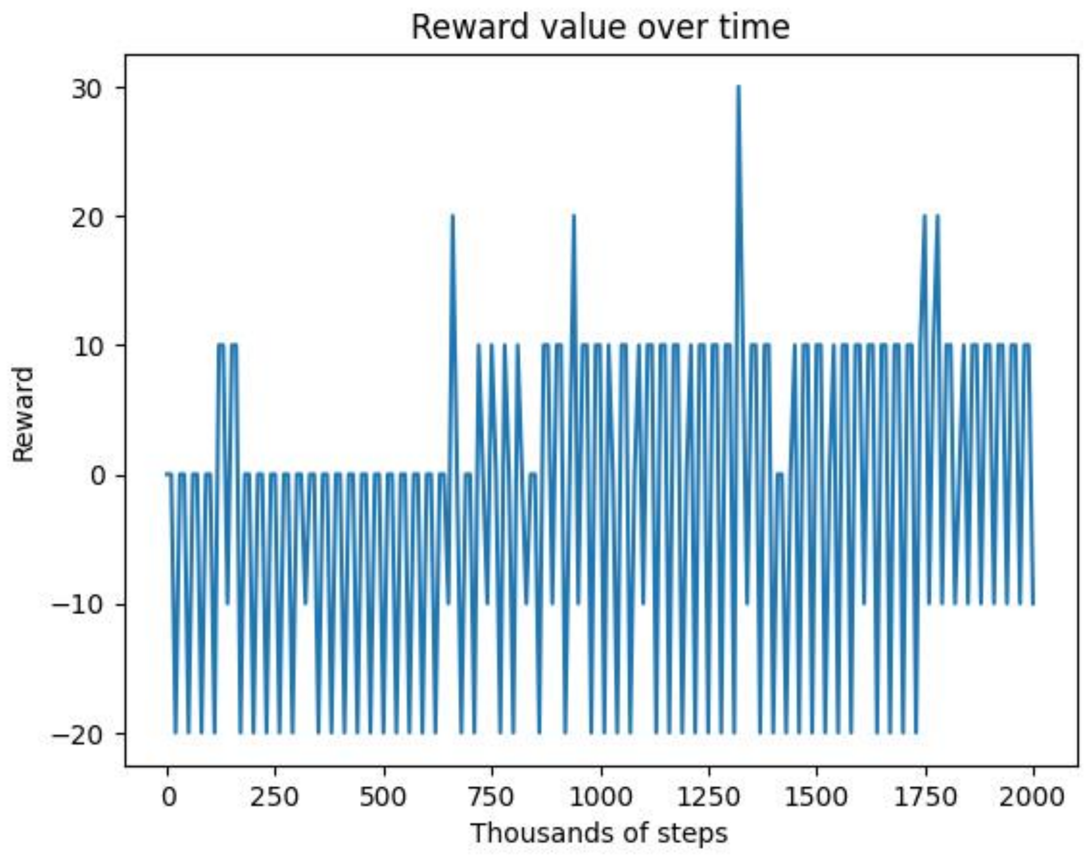
Adding level incentive only resulted in needless jumping.

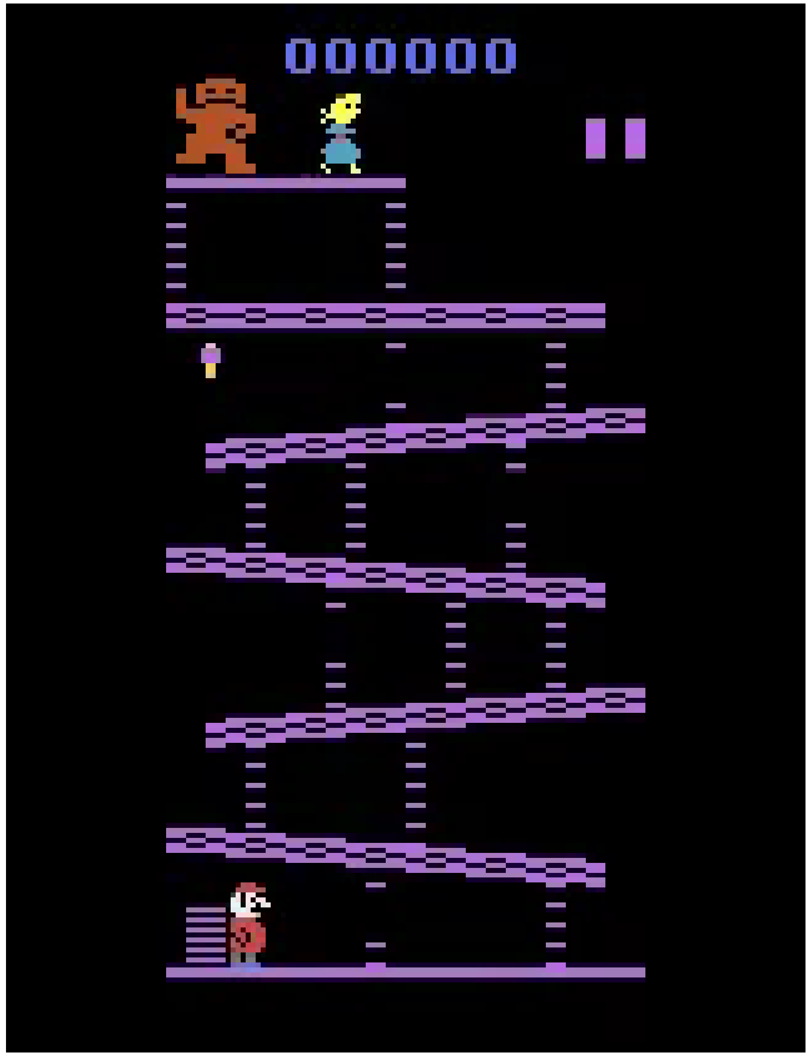




Action heuristics

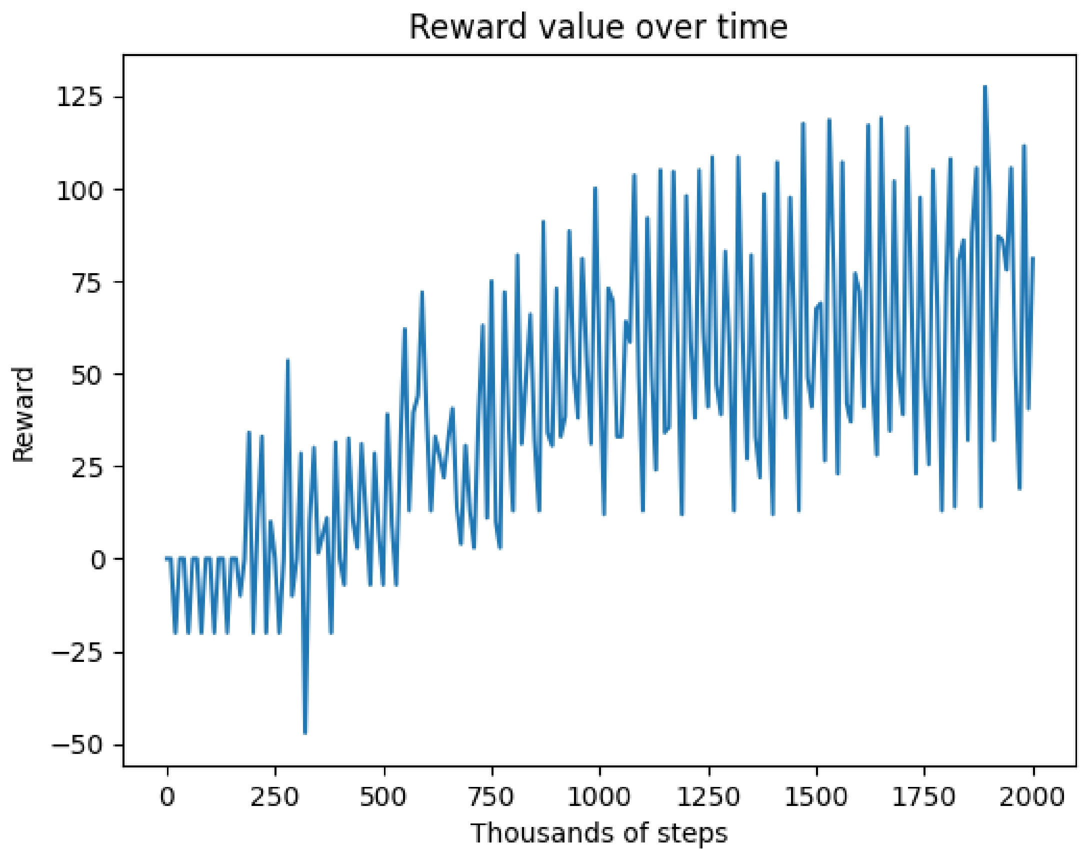
Adding action heuristics only resulted in agent doing random things.

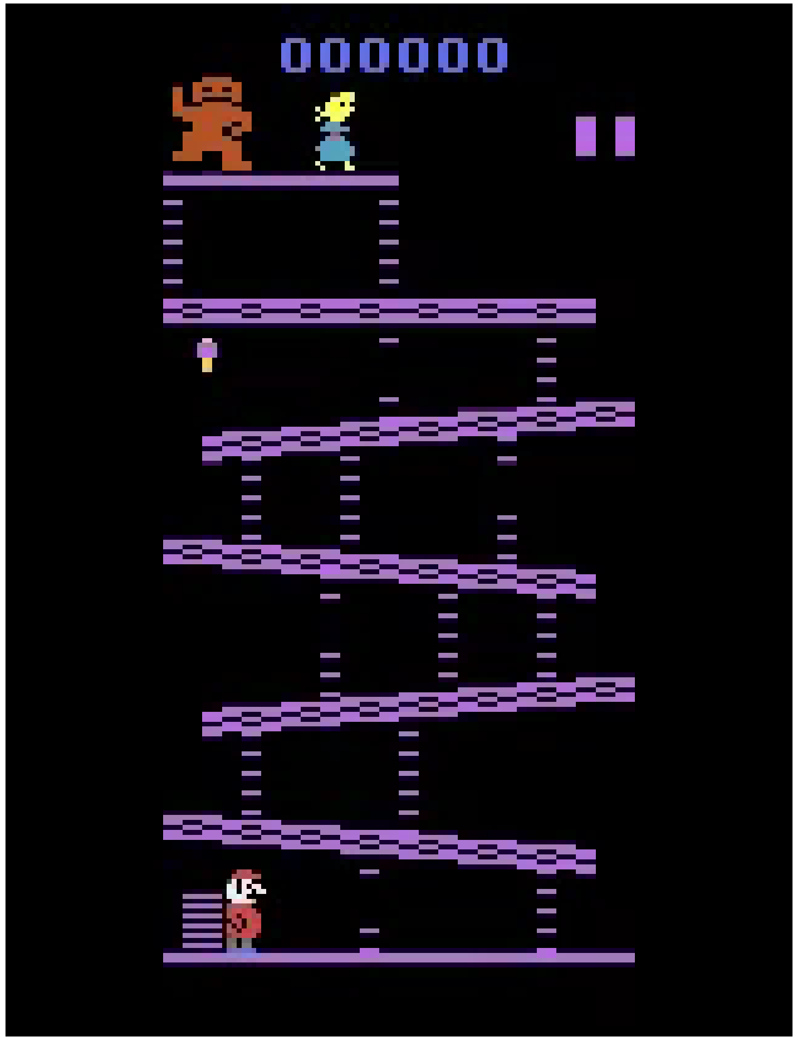




LAD + LVL

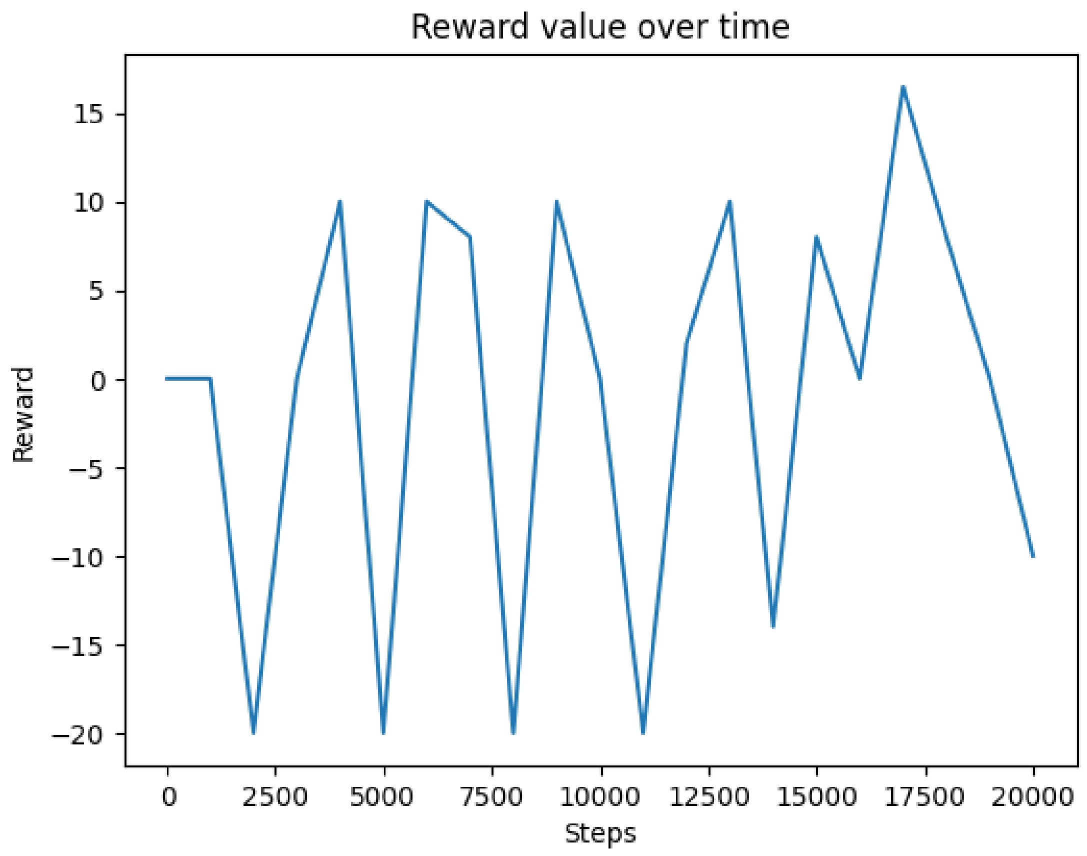
Combining ladders and level incentives resulted in the agent actually climbing to the fifth level.

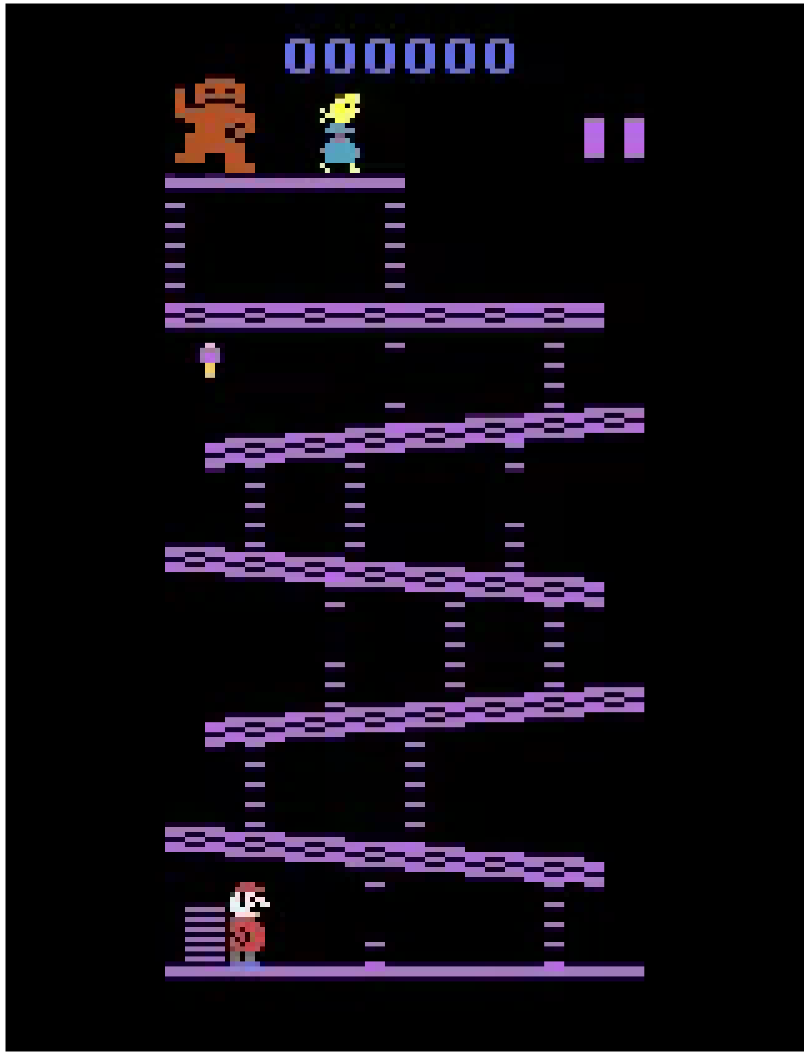




LAD + LVL + MS

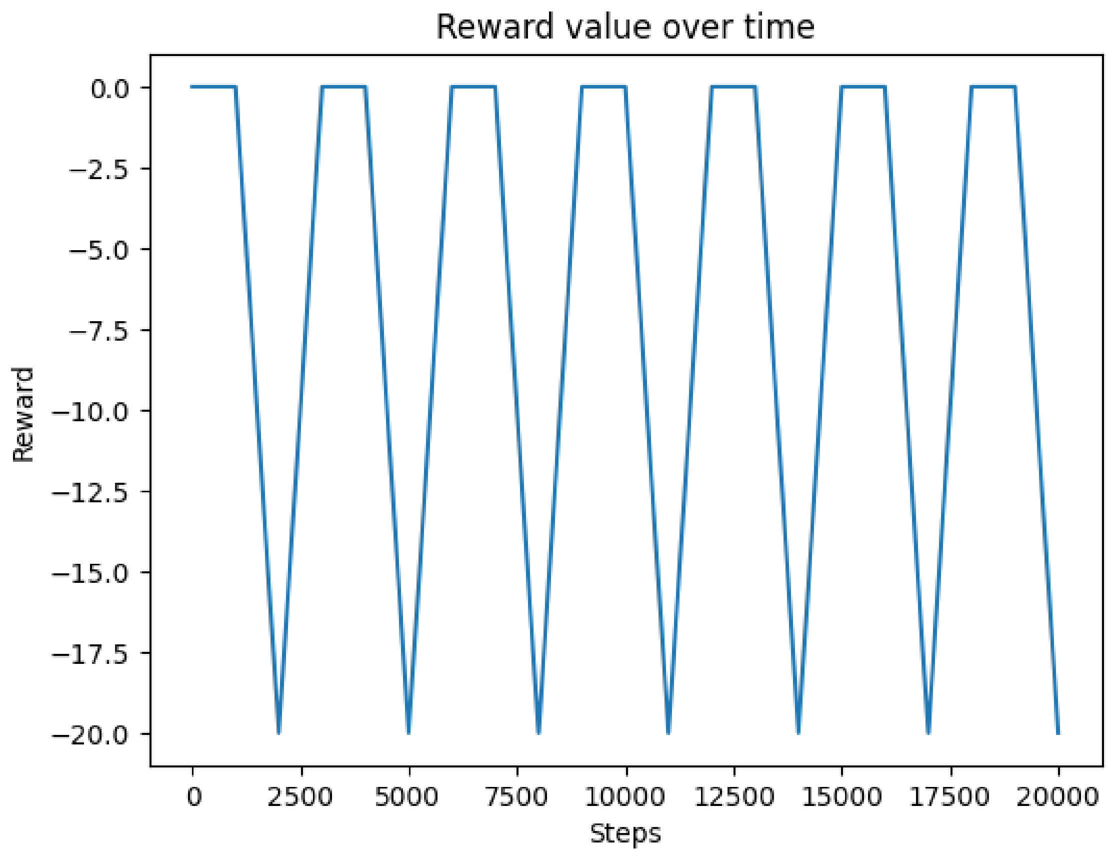
Combining all the incentives resulted in the agent going only right or standing/jumping without changing place.

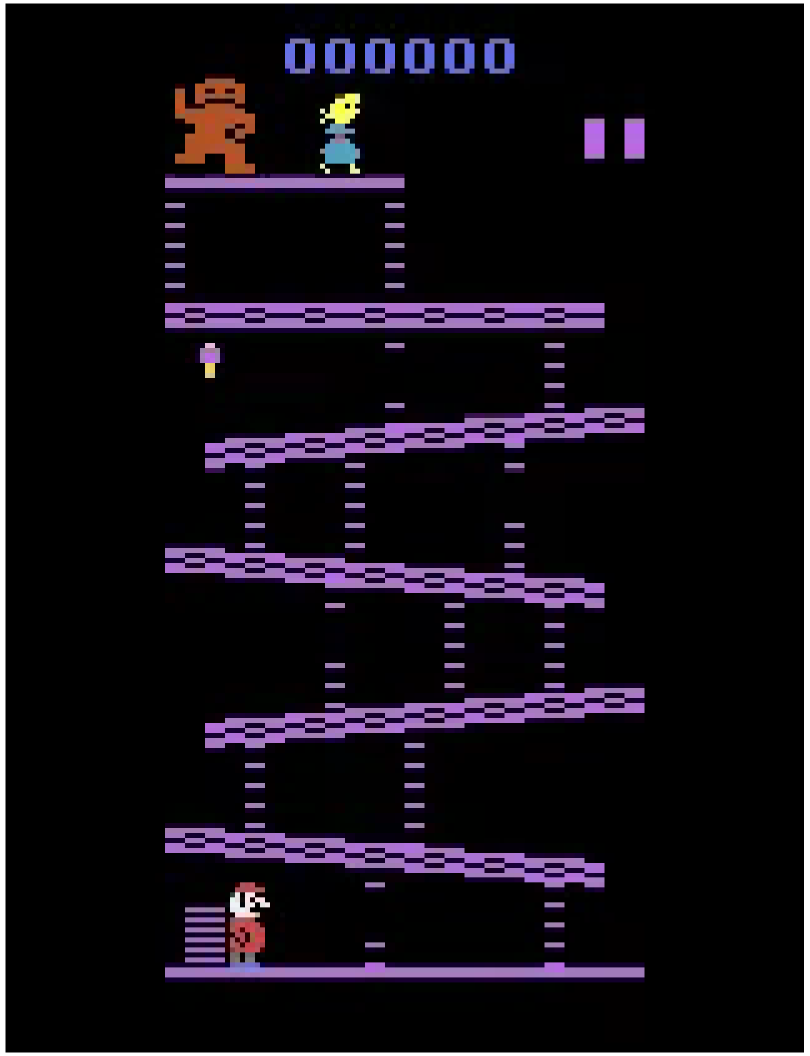




NJ + AH

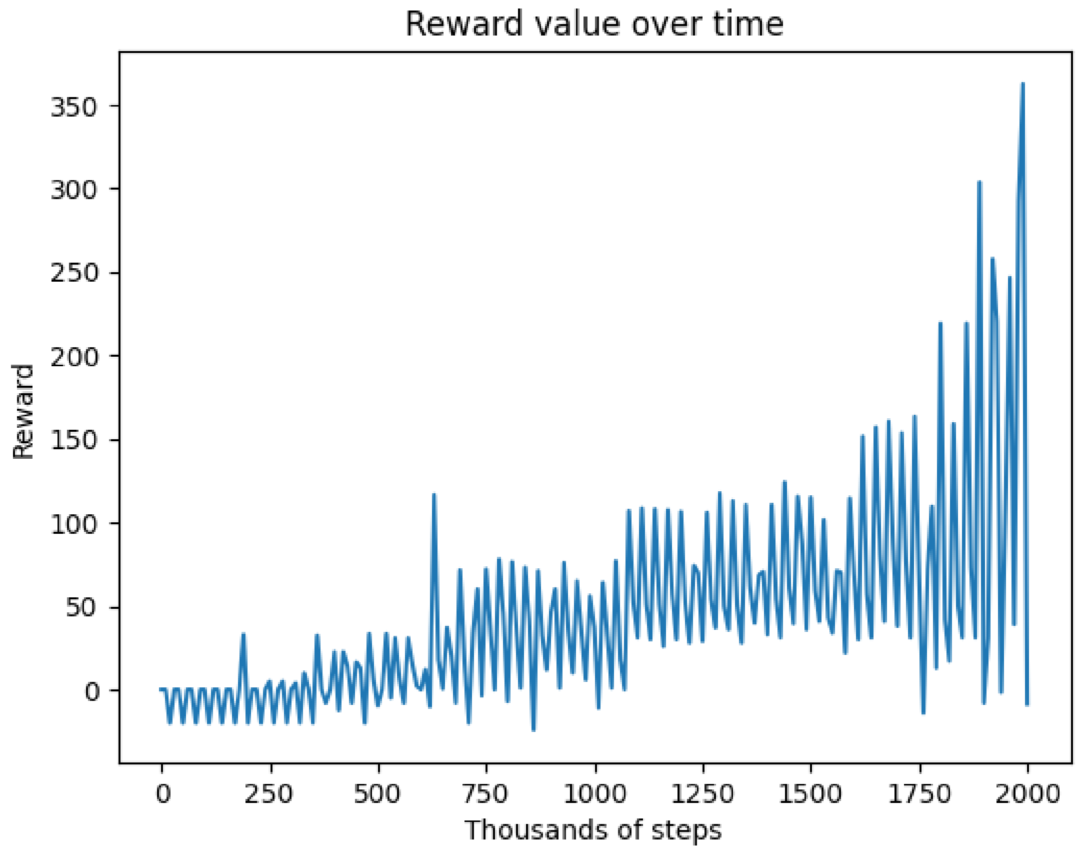
Adding the combination of punishing needless jumping and action heuristics resulted in the agent going in circles or standing still.

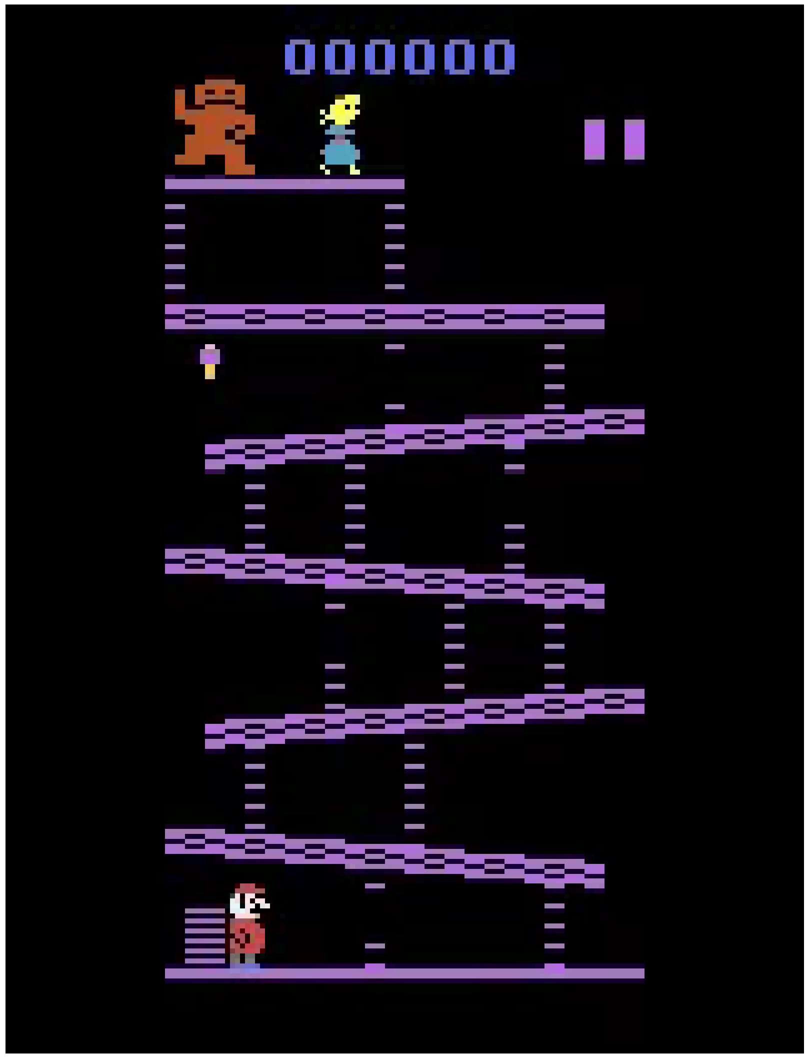




LAD + LVL+ PNJ

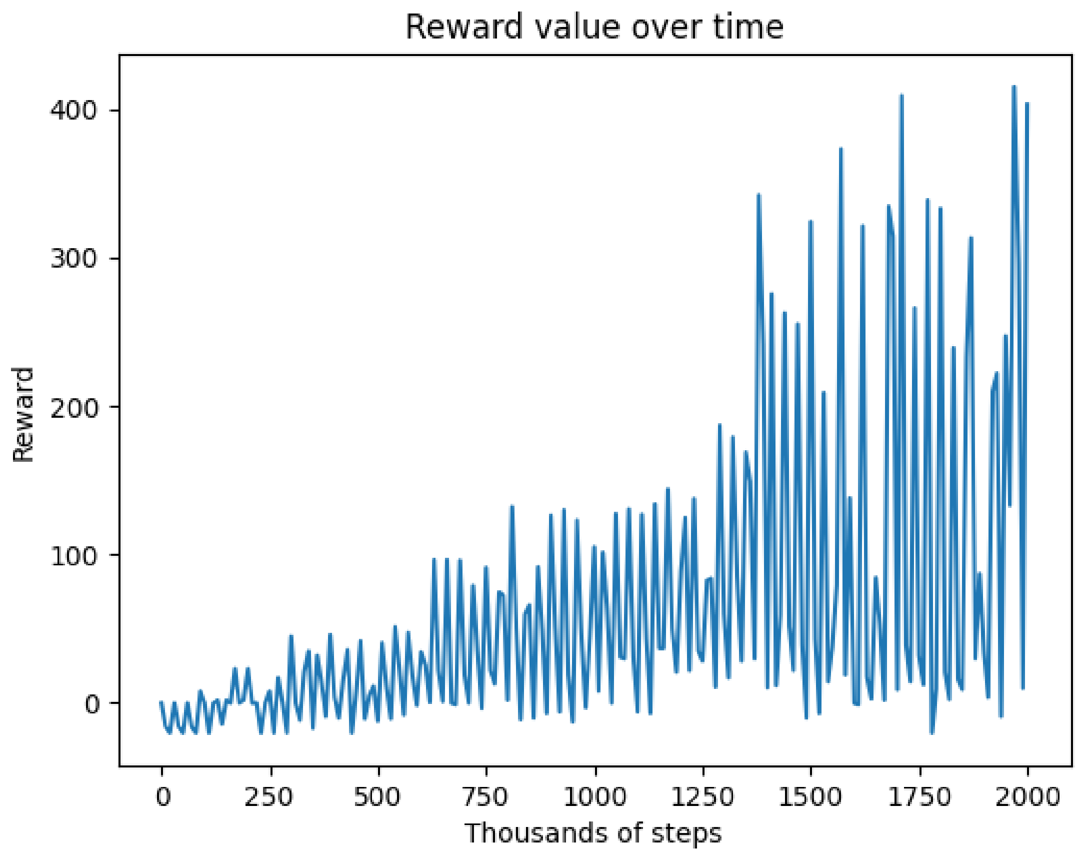
Combining ladders and level incentive with punishing needles jumping resulted in the agent finally climbing almost all the way up and getting the hammer.

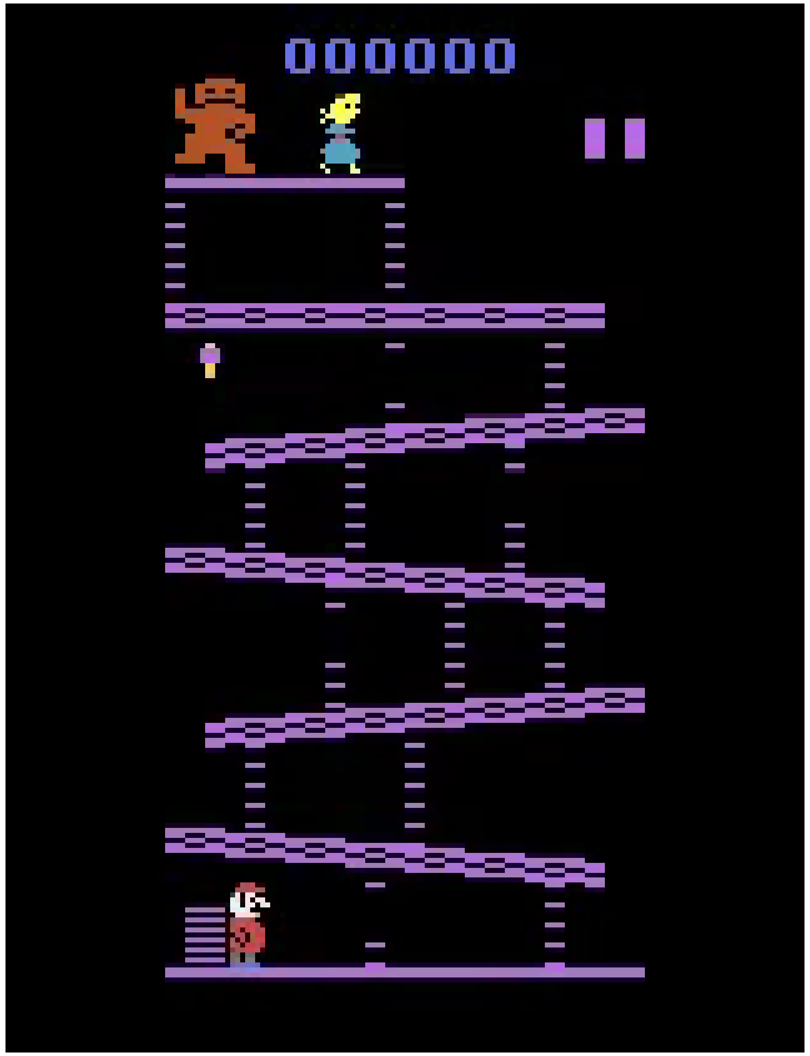




LAD + LVL + MS + PNJ

Combining all incentives with punishing needles jumping gave similar results.





Final results

Total steps: 2M

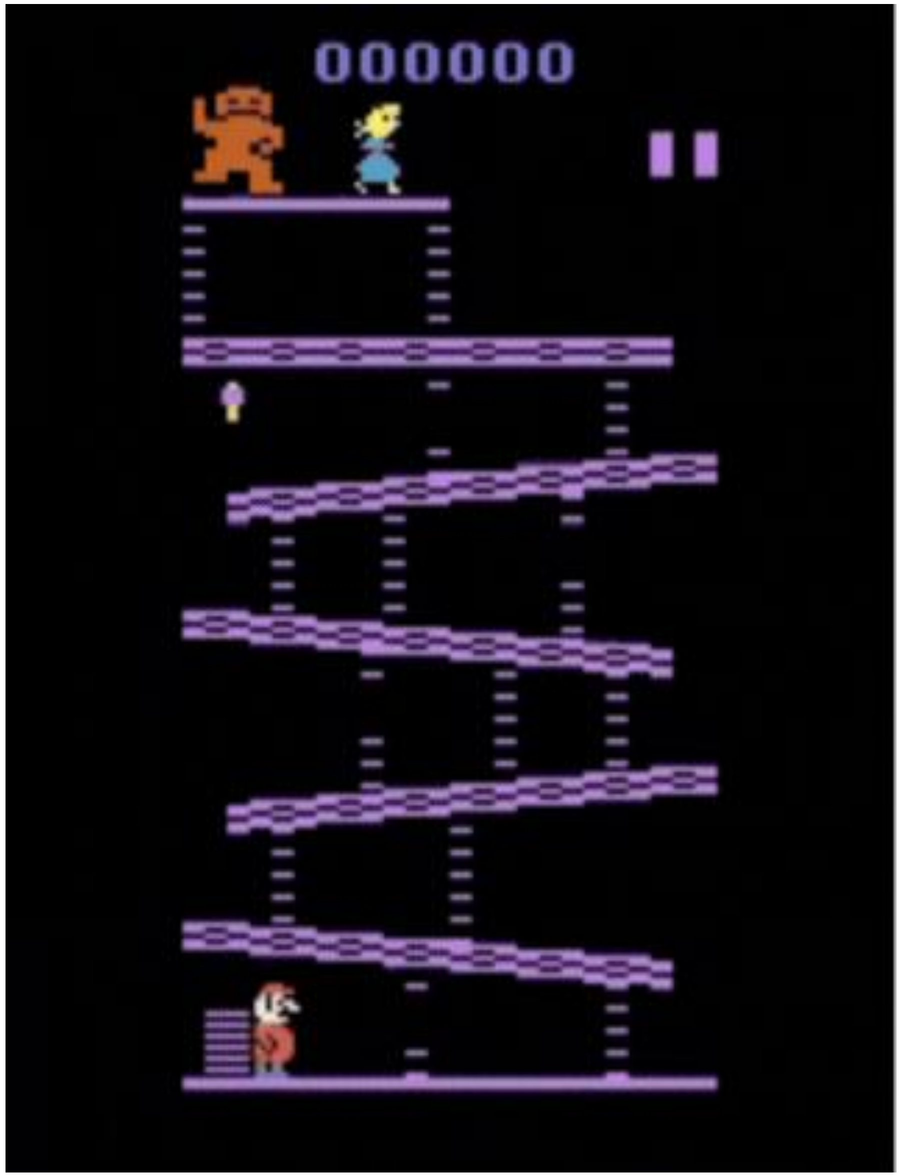
Gamma: 0.99

Linear schedule with epsilon going from 1 to 0.04 at 30% of steps

Modifications:

LAD + LVL + MS + PD + PNJ + AH

Training time: 12h



Bibliography

- [1] https://sportshub.cbsistatic.com/i/2023/09/26/b4f08ca9-d46d-40db-affe-6940c10aa551/donkey-kong.jpg
- [2] https://gymnasium.farama.org/environments/atari/donkey_kong/
- [3] https://gymnasium.farama.org/
- [4] https://github.com/DLR-RM/stable-baselines3
- [5] https://atariage.com/manual_html_page.php?SoftwareLabelID=149
- [6] https://github.com/torch/torch7/blob/master/README.md
- [7] https://opencv.org/

Thank you

- GitHub repository:

https://github.com/OptimalAgents/RL-DonkeyKong