CS 4200 Final Project

Aryan Miriyala

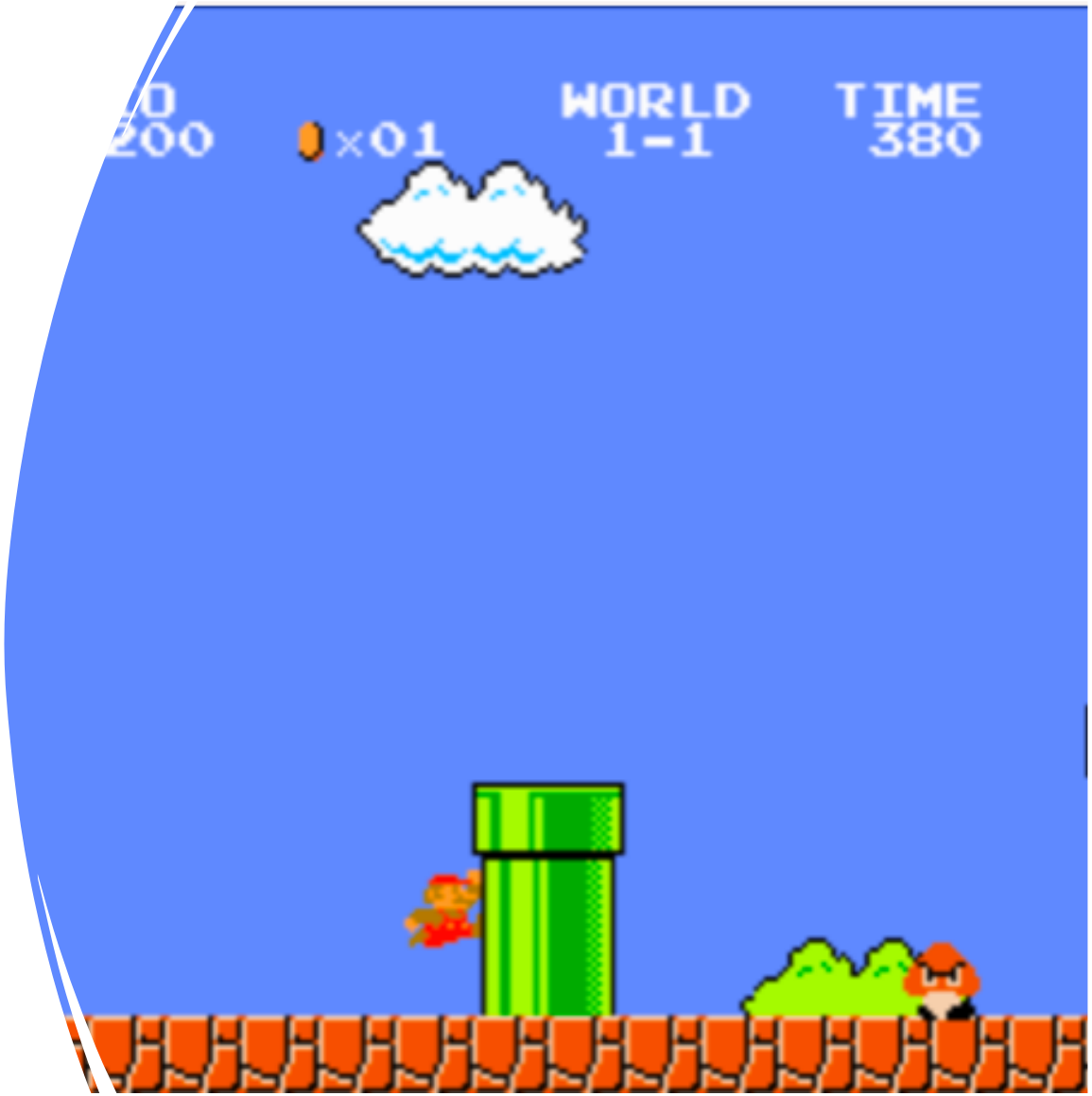
Cedric Dortch

Super Mario Reinforcement Learning Agent

Group 16

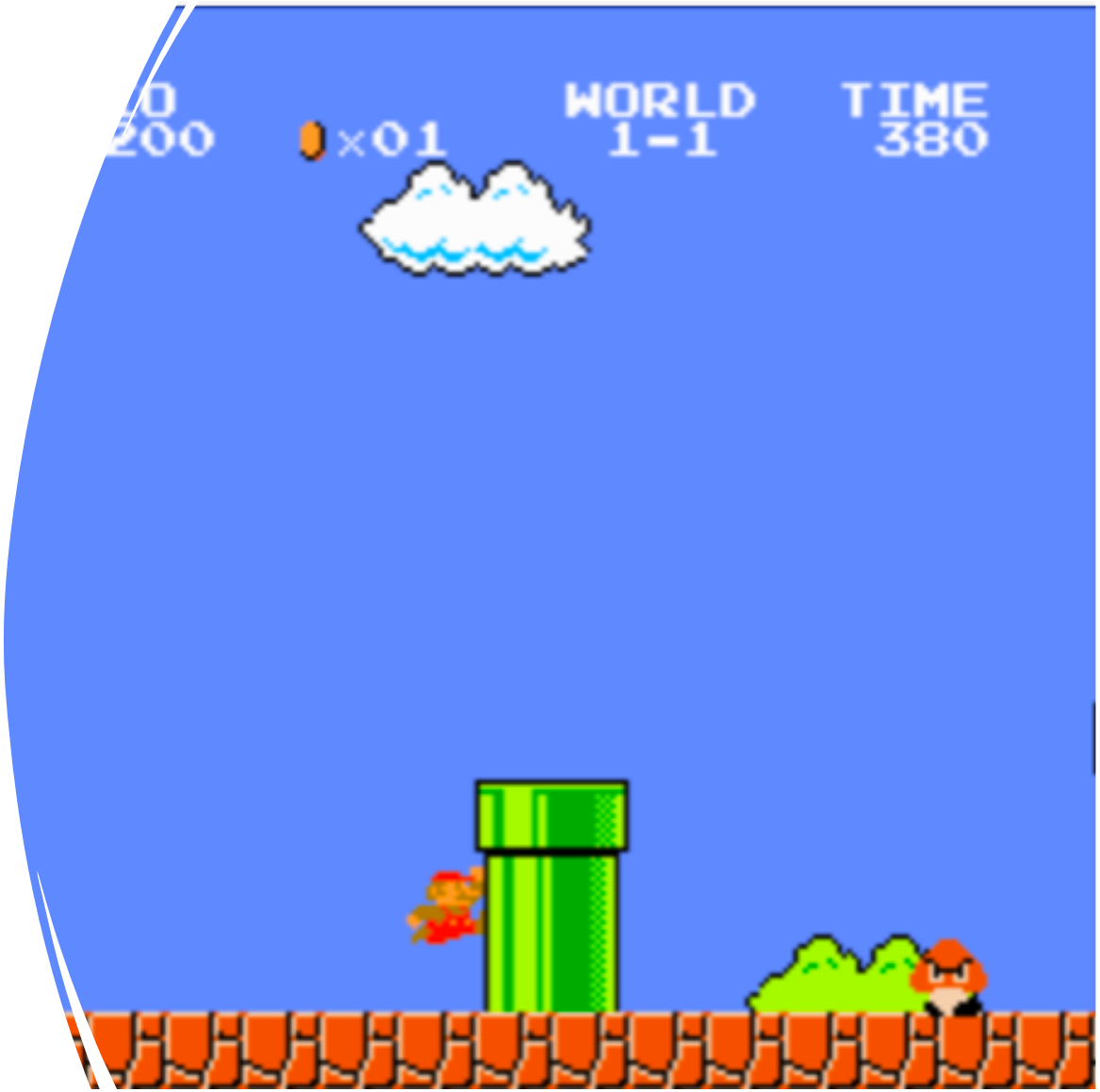
Contents in this Presentation

- Why Super Mario?
- Defining the Environment
- Reinforcement Learning
- Proximal Policy Optimization
- Deep Q-Network
- Results
- Challenges
- Conclusion



Why Super Mario?

- Popular game and an interesting topic in the context of AI
- Dynamic environment with tons of strategies
- The game is easily accessible through OpenAI Gym
- Extensive studies have been made on Super Mario Ai



Defining the environment

- Super Mario World PEAS representation
  - Performance: Reach the end of the level
  - Environment: The game world, the enemies, platforms, power ups
  - Actuators: The actions that are taken by Mario. We are restricted to SIMPLE_MOVEMENTS this time.
  - Sensors: Elements that provide information about the current state, position of Mario, obstacles and enemies.
- Action Space is restricted to cut down the amount of data that needs to be consumed
- A technique called gray scaling is used to cut down the data to be processed to 1/3rd

Defining the environment

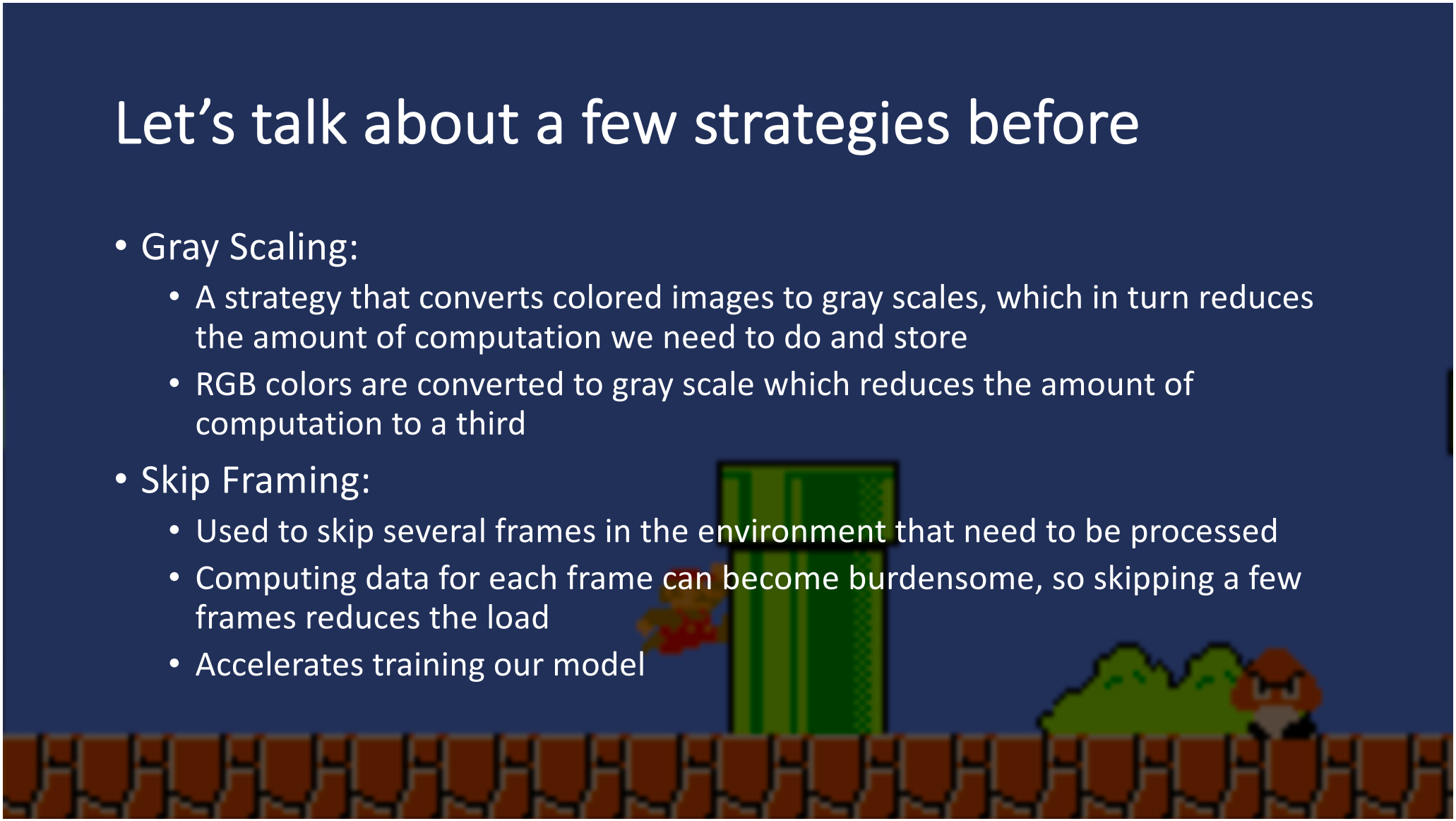
- We utilized the concept of wrapping our environment in different contexts to make use of several tricks to train our model
- Super Mario has a huge state space and action space, so brute force would take forever to converge
- Learning rate: Telling our model how important the new information learned should be. We take a really low number for a gradual convergence to the optimal value
- Gray Scaling, Frame Stacking, Resize Wrapping and Skip Framing are some concepts we are going to employ

Reinforcement Learning – Our Approach

- Learning how to maximize expected rewards based on observed transitions
- A machine learning approach to help software make decisions
- Imagine a Markov Decision Process, but we have no idea what the transition probabilities and reward functions are
- The agent explores the environment within the actions provided and learns what the optimal actions for a state are to maximize the reward
- We are going to talk about two popular algorithms that deal with reinforcement learning, especially when it comes to game training agents – Proximal Policy Optimization and Deep Q-Network

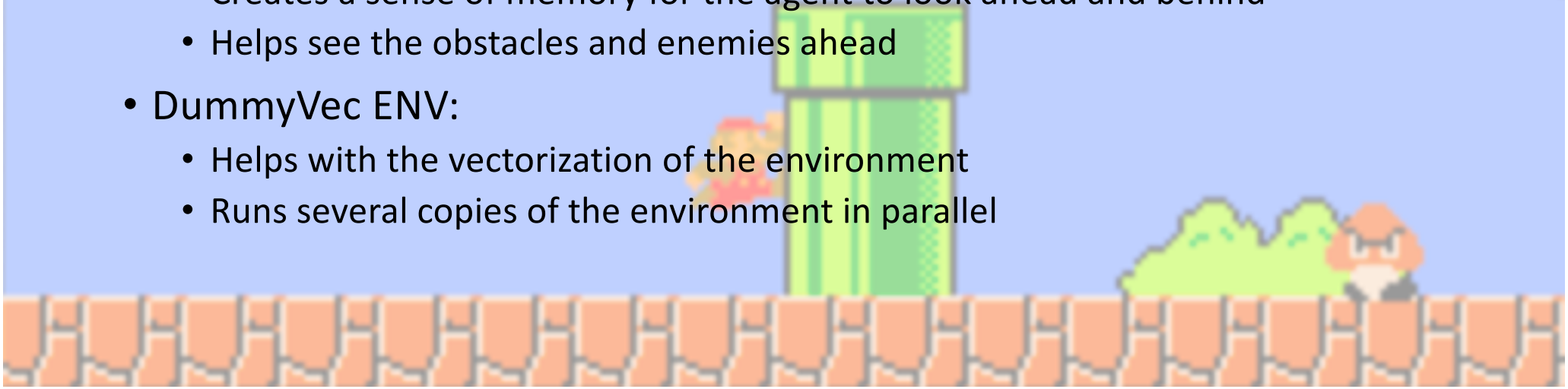
Let’s talk about a few strategies before

- Gray Scaling:
  - A strategy that converts colored images to gray scales, which in turn reduces the amount of computation we need to do and store
  - RGB colors are converted to gray scale which reduces the amount of computation to a third
- Skip Framing:
  - Used to skip several frames in the environment that need to be processed
  - Computing data for each frame can become burdensome, so skipping a few frames reduces the load
  - Accelerates training our model



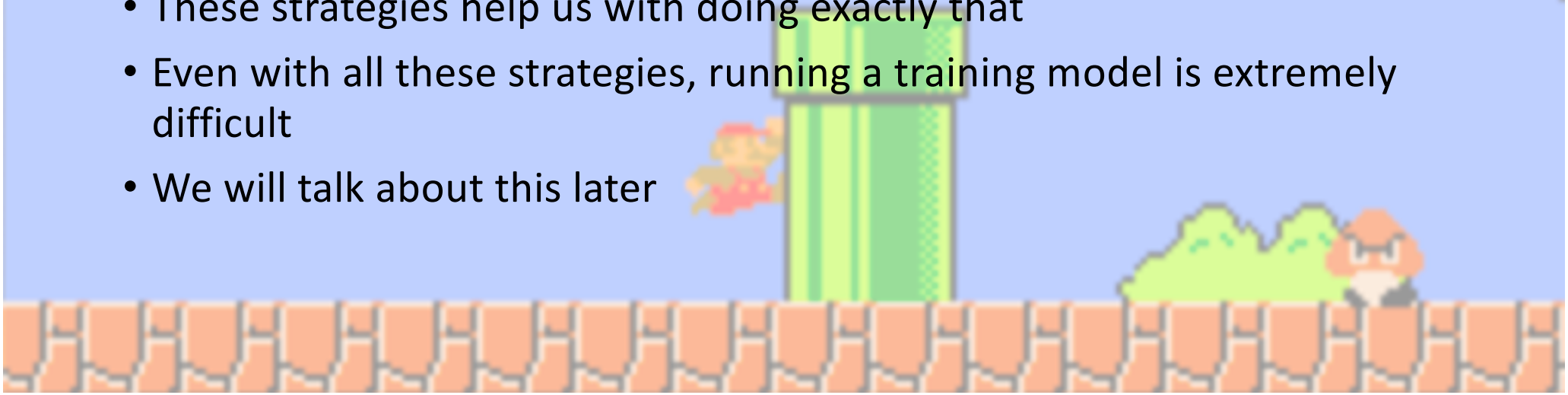
Let’s talk about a few strategies before

- Resize Operation:
  - Resize the resolution of the frames processed to reduce data size
- Frame Stacking:
  - Stacks multiple frames together for multi framed observation
  - Creates a sense of memory for the agent to look ahead and behind
  - Helps see the obstacles and enemies ahead
- DummyVec ENV:
  - Helps with the vectorization of the environment
  - Runs several copies of the environment in parallel



Why talk about these concepts?

- The computation we will be performing to achieve a model close to optimal convergence is very high
- Only way to do it with the resources available is to utilize strategies that reduce the amount of data that we look at
- These strategies help us with doing exactly that
- Even with all these strategies, running a training model is extremely difficult
- We will talk about this later

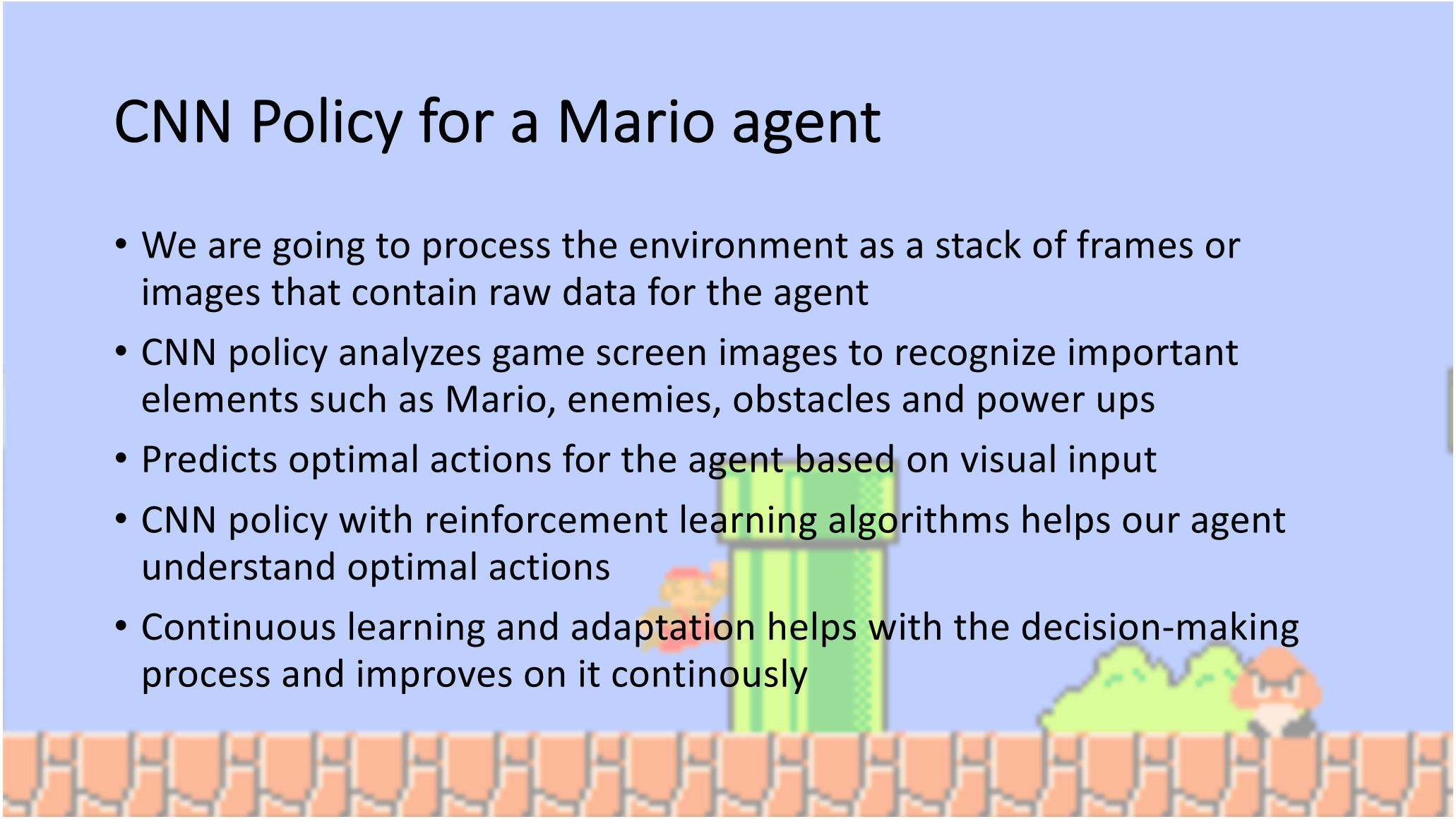


CNN Policy (Convoluted Neural Network)

- A specialized neural network architecture for analyzing visual data
- Utilized convoluted layers to detect features such as the environment that it is in, the hurdles and the enemies
- The policy automatically extracts from raw visual data
- Very helpful for interpreting image frames, such as when gaming
- Helps decision making for agents that are navigating complex environments

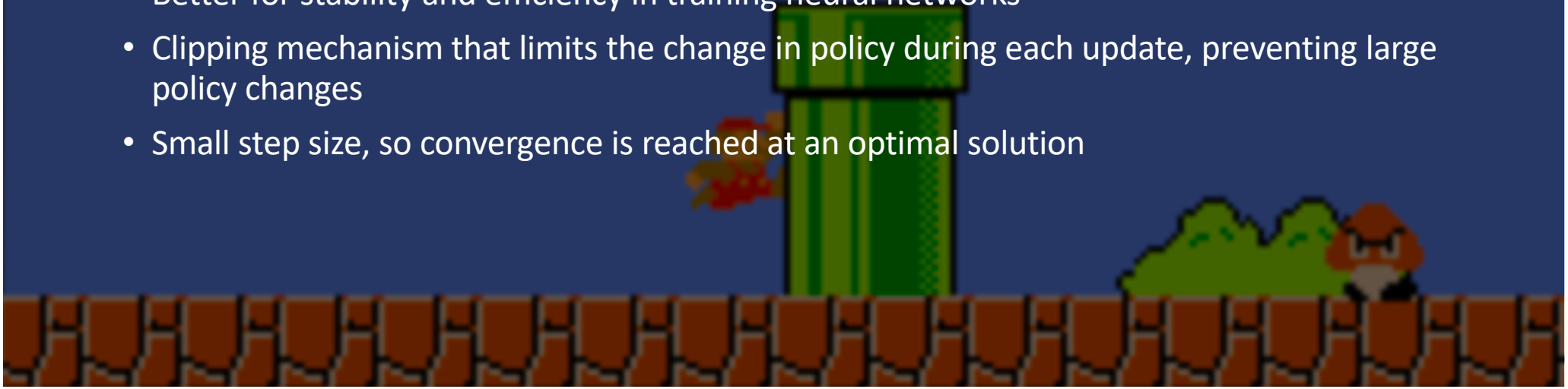
CNN Policy for a Mario agent

- We are going to process the environment as a stack of frames or images that contain raw data for the agent
- CNN policy analyzes game screen images to recognize important elements such as Mario, enemies, obstacles and power ups
- Predicts optimal actions for the agent based on visual input
- CNN policy with reinforcement learning algorithms helps our agent understand optimal actions
- Continuous learning and adaptation helps with the decision-making process and improves on it continously

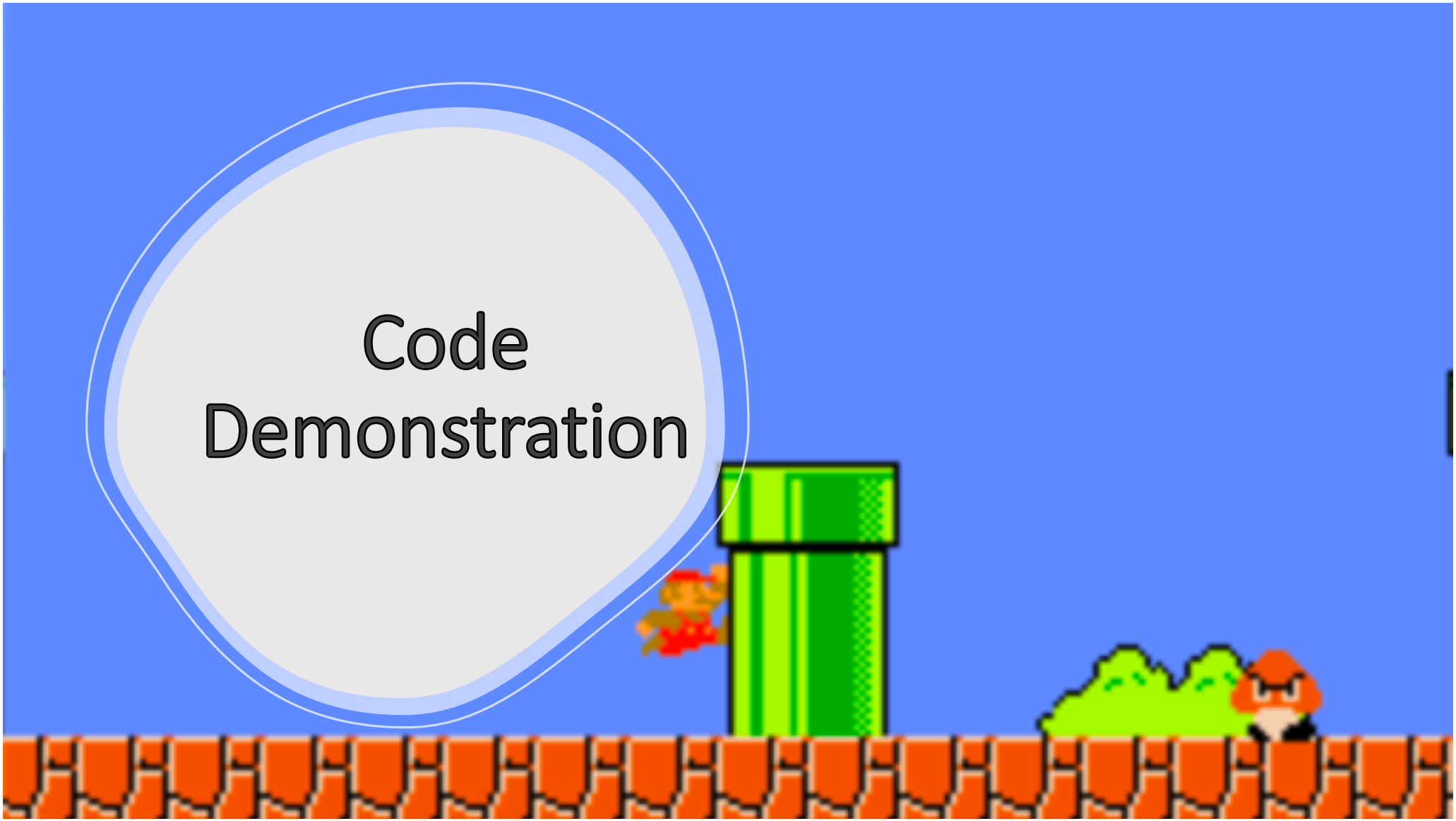


Proximal Policy Optimization (PPO)

- Reinforcement Learning algorithm that trains agents to performs tasks where the actions impact future states and rewards
- Focuses on optimizing the policy function mapping observed states to relevant actions
- Iterative updates helps the agent learn to make better decisions, optimizing over time
- Better for stability and efficiency in training neural networks
- Clipping mechanism that limits the change in policy during each update, preventing large policy changes
- Small step size, so convergence is reached at an optimal solution



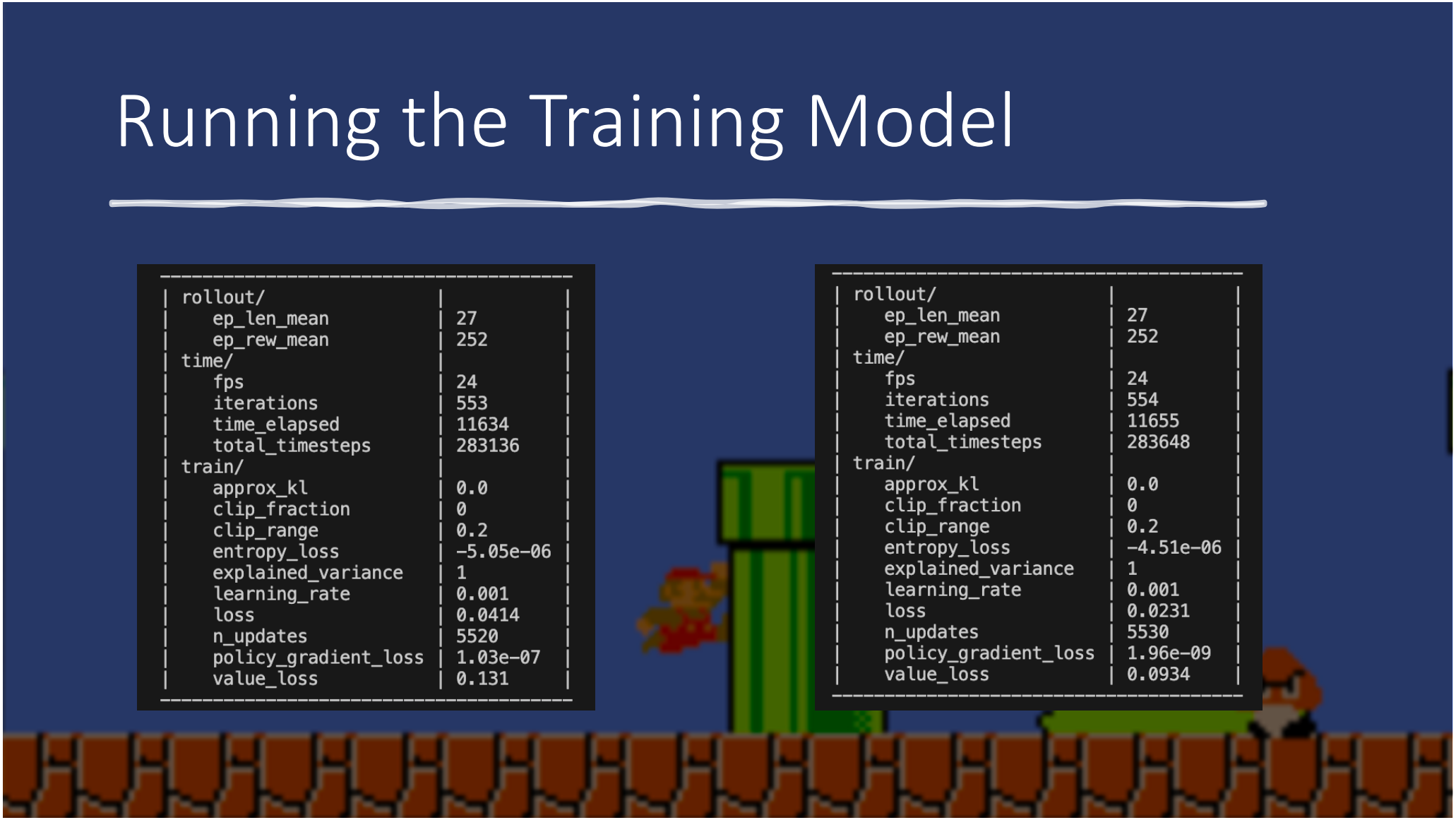
Code Demonstration



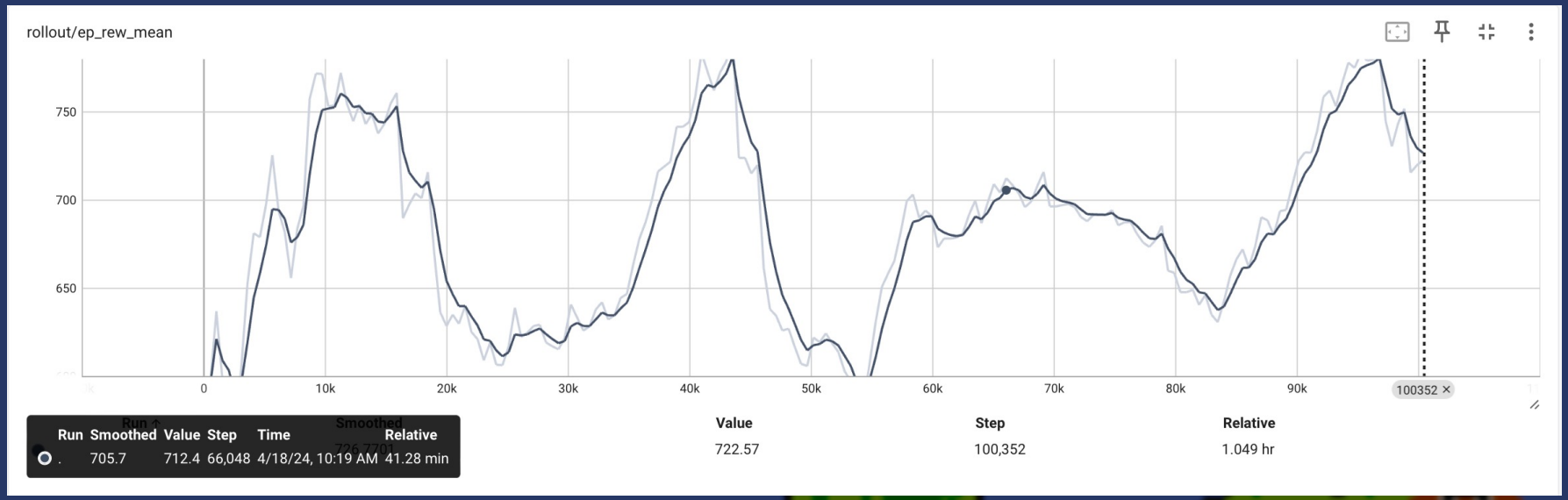
Running the Training Model

rollout/	
ep_len_mean	27
ep_rew_mean	252
time/	
fps	24
iterations	553
time_elapsed	11634
total_timesteps	283136
train/	
approx_kl	0.0
clip_fraction	0
clip_range	0.2
entropy_loss	-5.05e-06
explained_variance	1
learning_rate	0.001
loss	0.0414
n_updates	5520
policy_gradient_loss	1.03e-07
value_loss	0.131

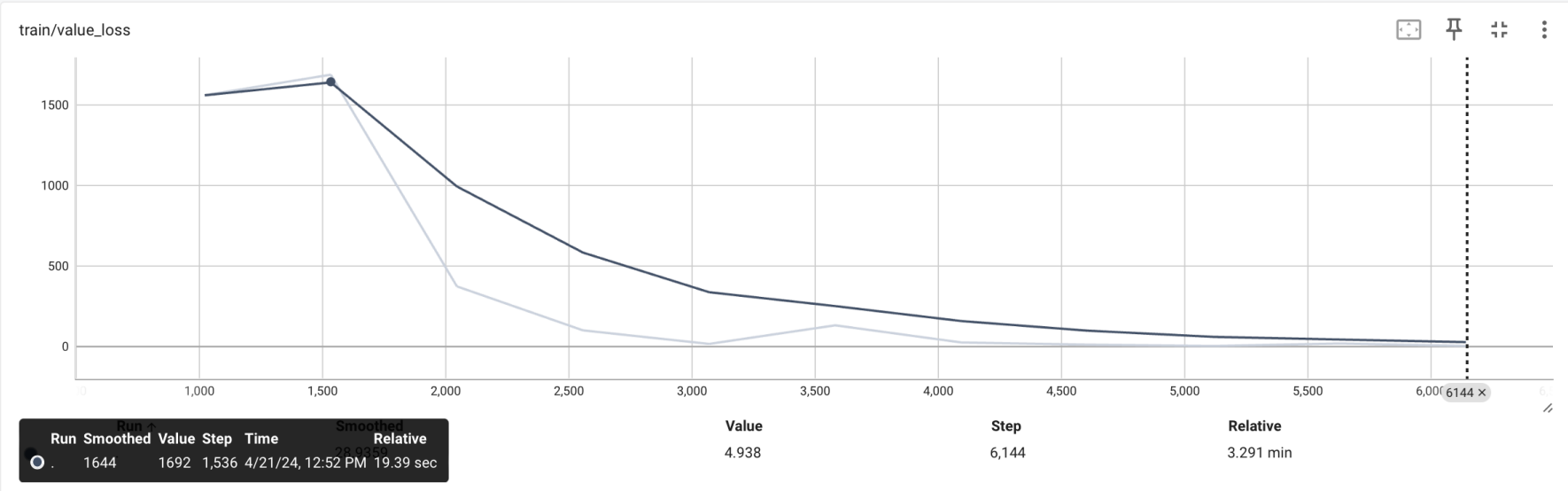
rollout/	
ep_len_mean	27
ep_rew_mean	252
time/	
fps	24
iterations	554
time_elapsed	11655
total_timesteps	283648
train/	
approx_kl	0.0
clip_fraction	0
clip_range	0.2
entropy_loss	-4.51e-06
explained_variance	1
learning_rate	0.001
loss	0.0231
n_updates	5530
policy_gradient_loss	1.96e-09
value_loss	0.0934



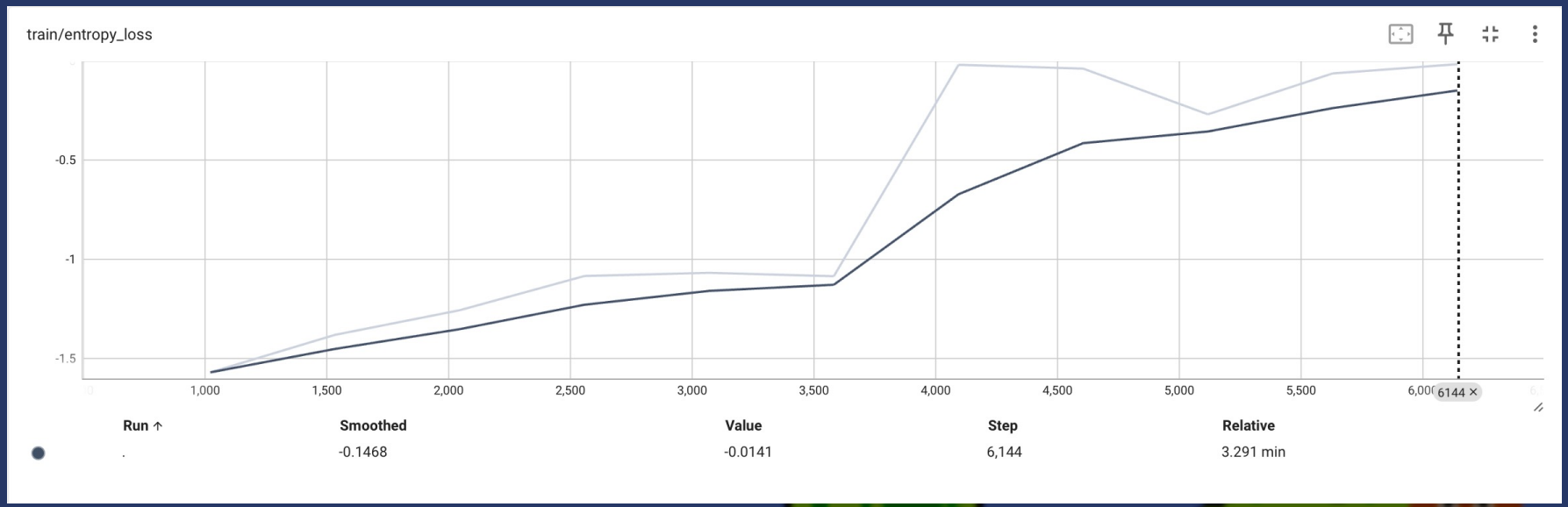
Let’s Talk Graphs – Episode Rewards



Let’s Talk Graphs – Value Loss

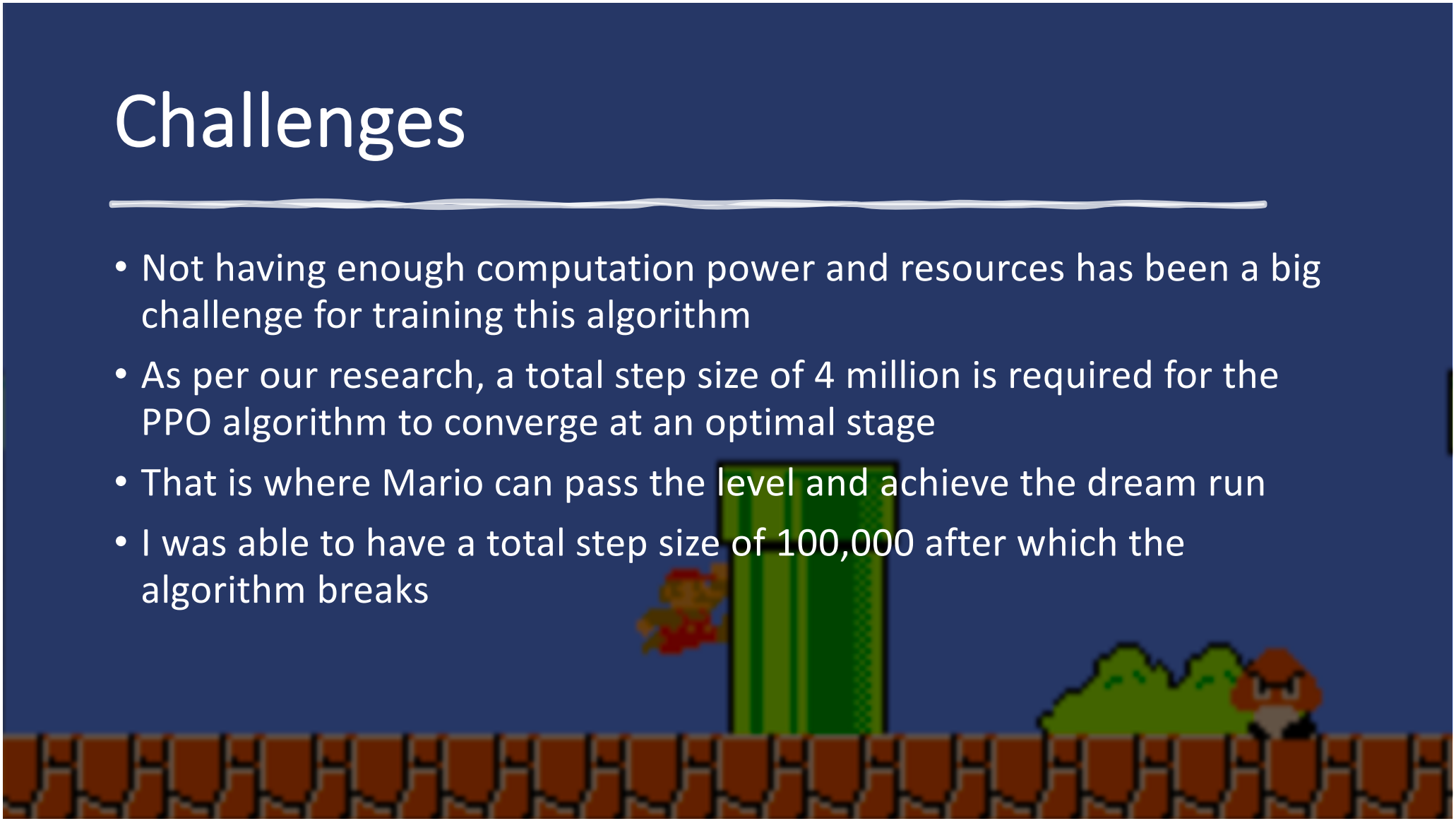


Let’s Talk Graphs – Entropy Loss



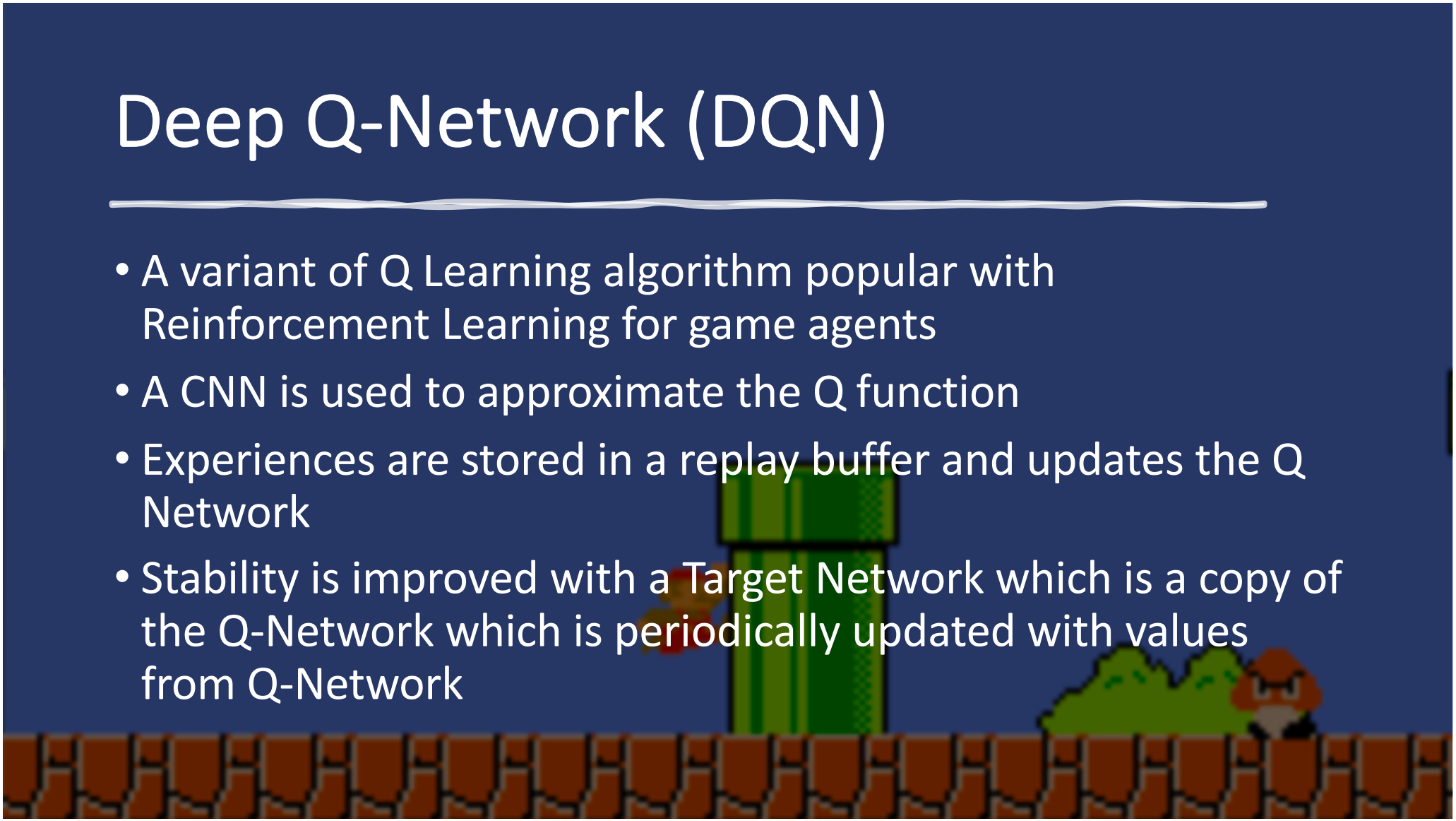
Challenges

- Not having enough computation power and resources has been a big challenge for training this algorithm
- As per our research, a total step size of 4 million is required for the PPO algorithm to converge at an optimal stage
- That is where Mario can pass the level and achieve the dream run
- I was able to have a total step size of 100,000 after which the algorithm breaks

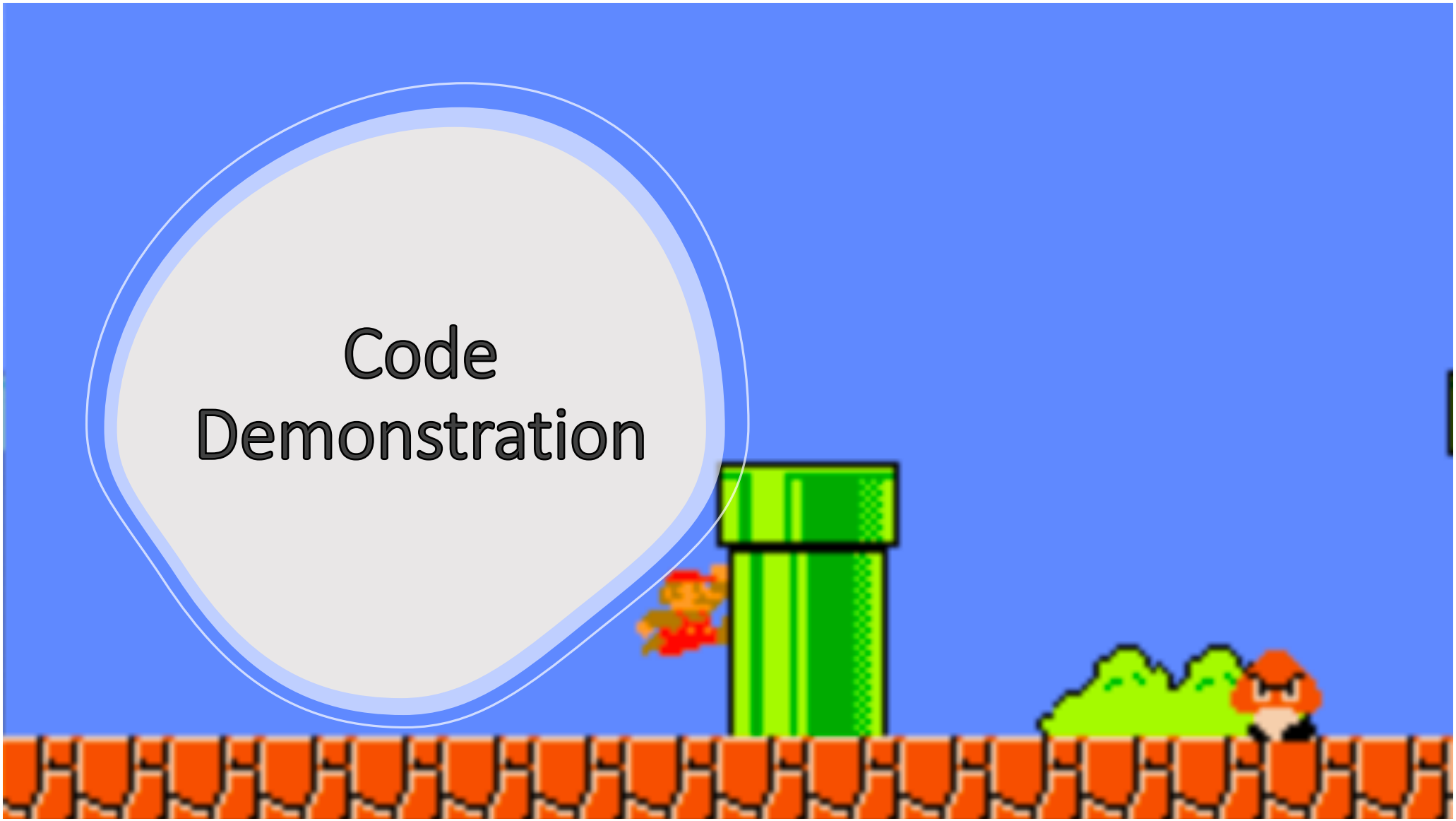


Deep Q-Network (DQN)

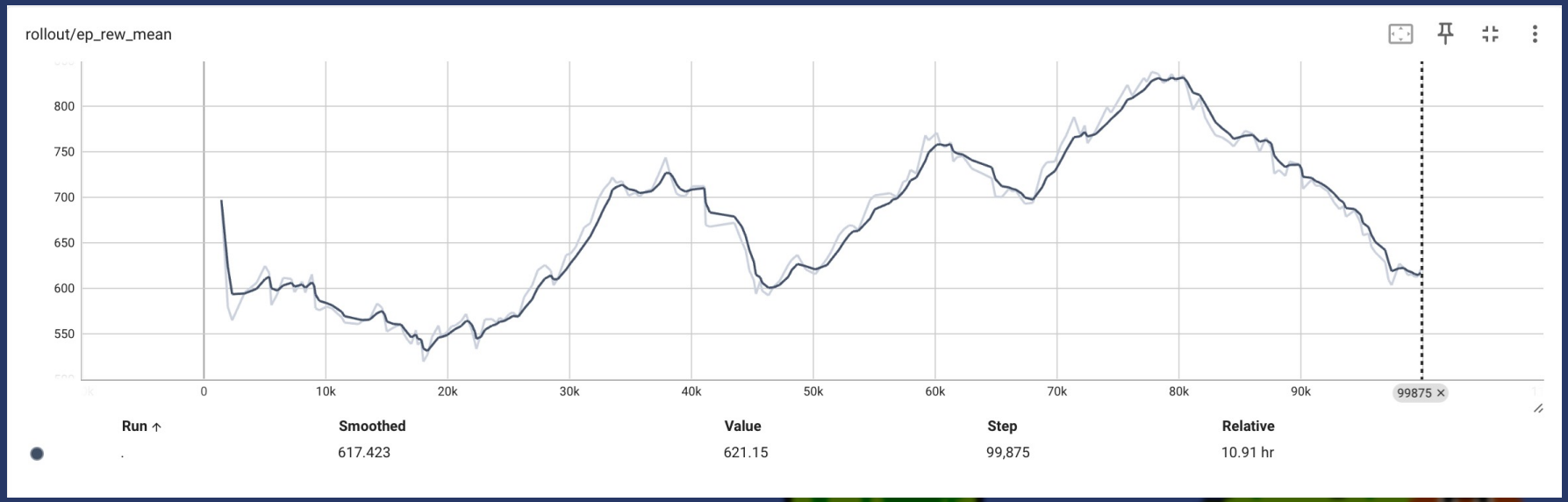
- A variant of Q Learning algorithm popular with Reinforcement Learning for game agents
- A CNN is used to approximate the Q function
- Experiences are stored in a replay buffer and updates the Q Network
- Stability is improved with a Target Network which is a copy of the Q-Network which is periodically updated with values from Q-Network



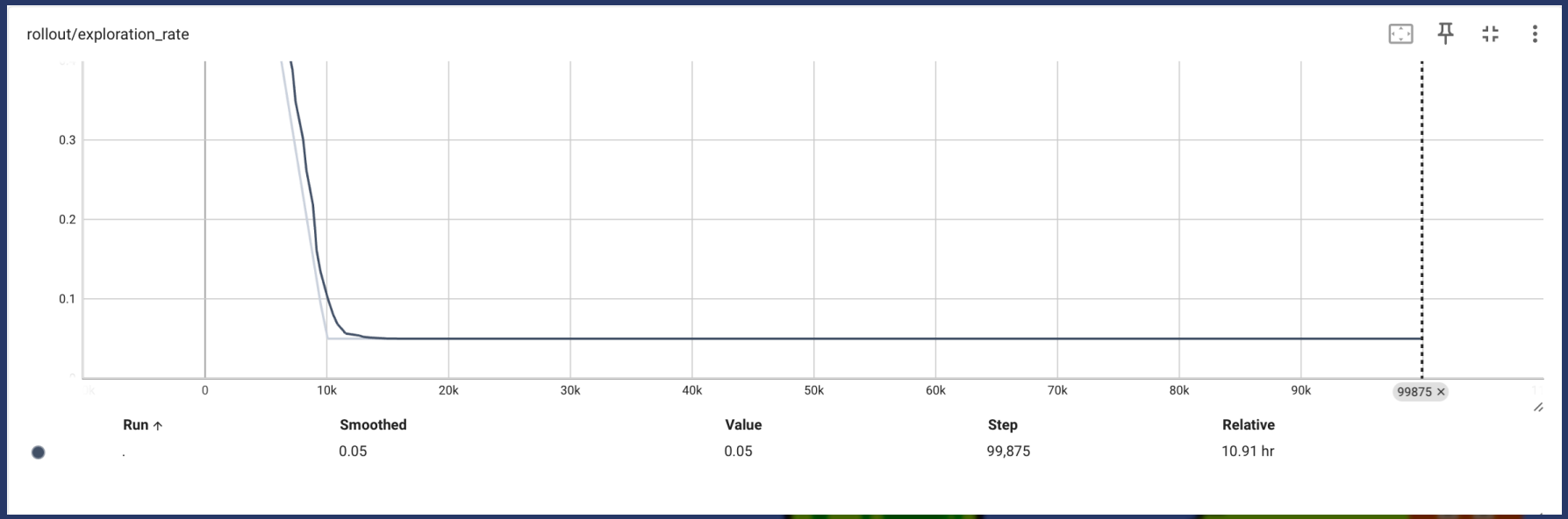
Code Demonstration



Let’s Talk Graphs – Episode Rewards

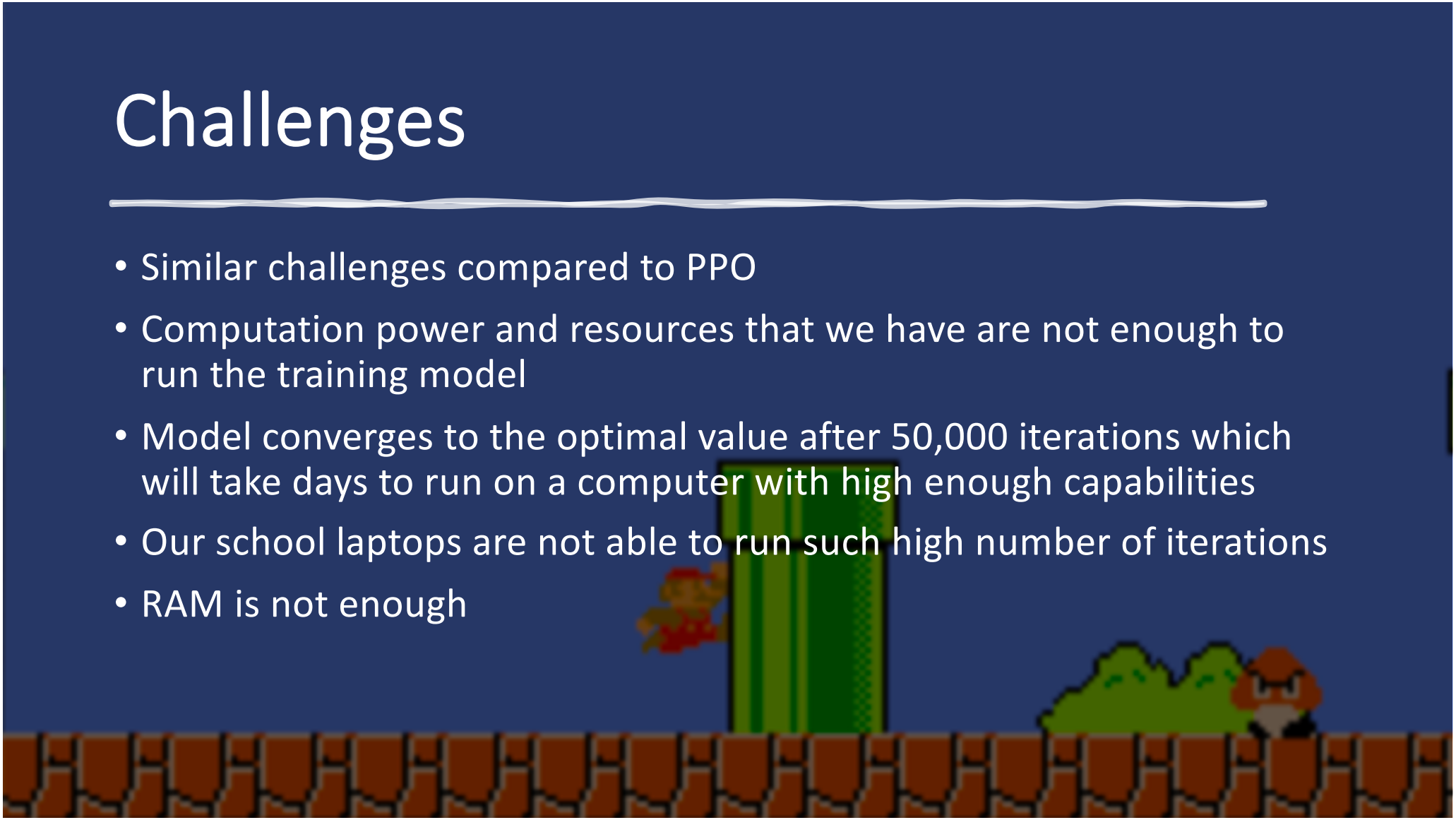


Let’s Talk Graphs – Exploration Rate



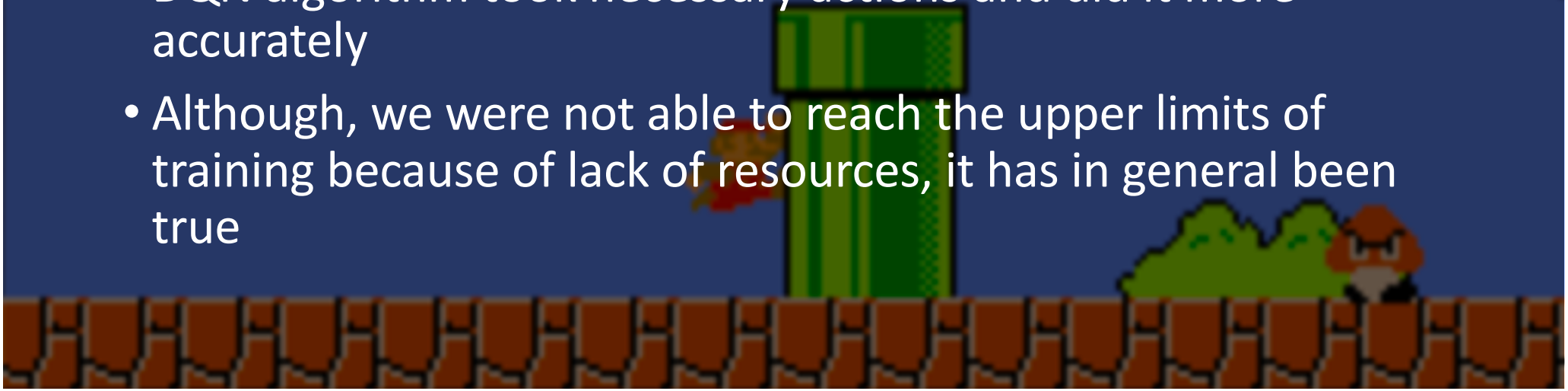
Challenges

- Similar challenges compared to PPO
- Computation power and resources that we have are not enough to run the training model
- Model converges to the optimal value after 50,000 iterations which will take days to run on a computer with high enough capabilities
- Our school laptops are not able to run such high number of iterations
- RAM is not enough



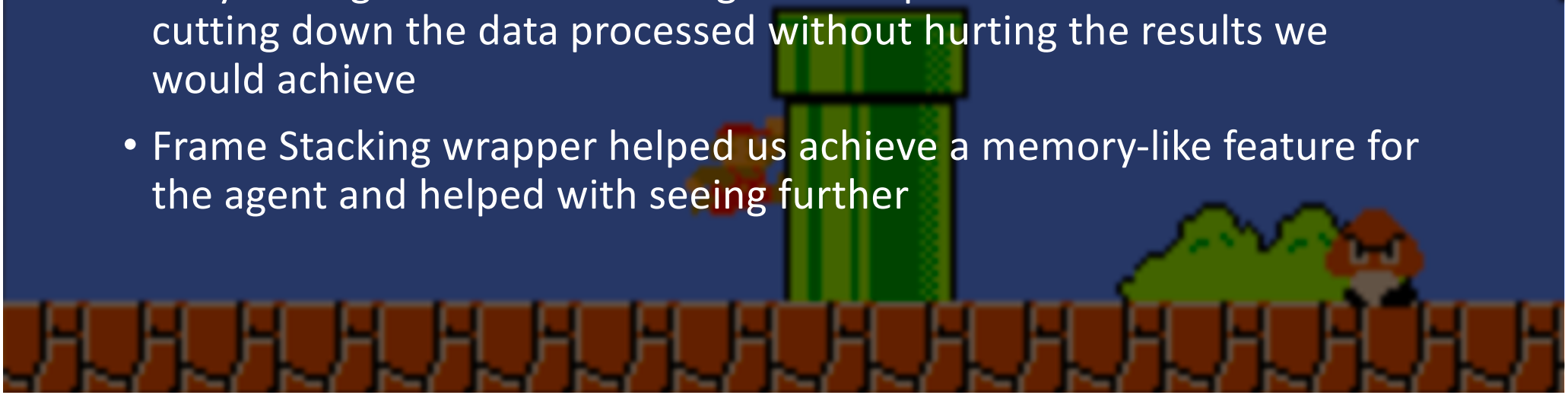
Conclusion - Lessons Learned – PPO vs DQN

- Theoretically DQN is supposed to perform better than PPO, and it seems like through our exploration this has been the case
- DQN algorithm took necessary actions and did it more accurately
- Although, we were not able to reach the upper limits of training because of lack of resources, it has in general been true



Conclusion – Wrapper Strategies

- Wrapping the environment and the agent within a few strategies has been very interesting
- We reduced the computational load for by about a half!
- Gray scaling and Frame Resizing have helped us achieve this idea of cutting down the data processed without hurting the results we would achieve
- Frame Stacking wrapper helped us achieve a memory-like feature for the agent and helped with seeing further



Conclusion – Overall

- We learned about numerous concept of reinforcement learning and how to evaluate rewards
- The reward function for Mario is very complex, considering how many different thing we need to consider
- It was designed by the writers of PPO and DQN respectively and the software for the reward function is owned by them
- Both the algorithms do a very good job at trying achieve convergence, DQN being the overall better, but this decision is heavily based on the action space chosen and the description of the environment