Implementation of Q-Learning and Deep Q-learning Algorithms

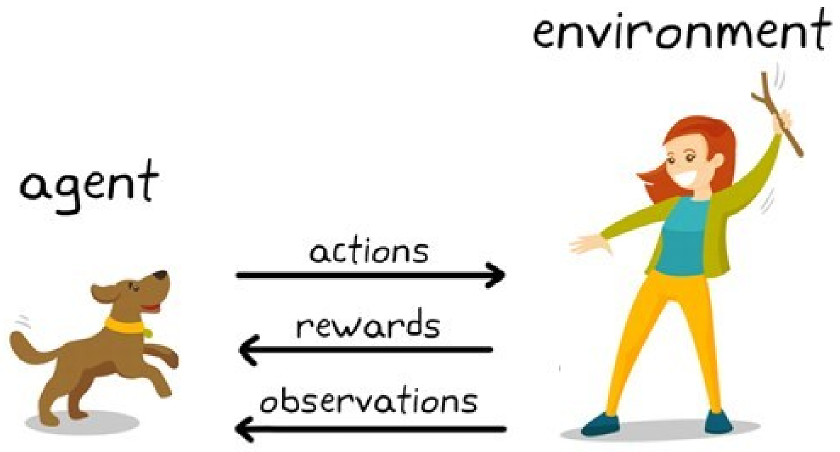
Göktuğ YILDIRIM
041502012
May 2020

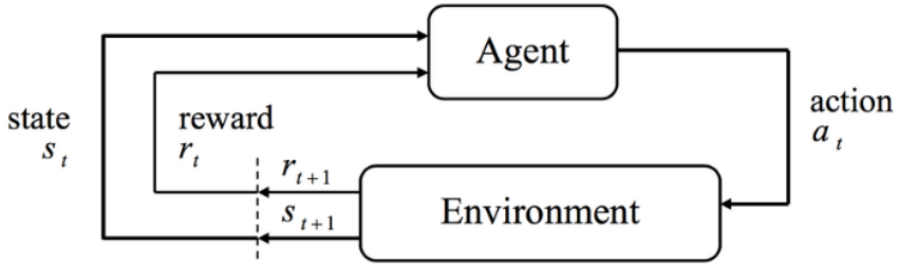
Outline

- Project Definition:
  - Training Reinforcement Learning Agent
  - Environments
- Methods:
  - Q-Learning Algorithm
    - Exploitation and Exploration
    - Parameters
    - Experimental Results and Tunning
  - Deep Q-Learning Algorithm
    - Parameters
    - Experimental Results and Tunning
- Conclusion
- References

Project Definition

- The goal of the project is to train and develop Reinforcement Learning agent with Q-Learning and Deep Q-Learning algorithms.
- What is Reinforcement Learning?
  - An approach to Artificial Intelligence
  - Reinforcement Learning is the training of machine learning models to make a sequence of decisions. Also, model is called agent.
  - The agent is one who takes decisions based on the rewards and punishments.
  - The goal of the agent is to select the best action to maximize rewards.





Images are taken from: https://www.kdnuggets.com/2019/10/mathworks-reinforcement-learning.html

https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

Environments

Environments

Taxi-v2

This task was introduced in [Dietterich2000] to illustrate some issues in hierarchical reinforcement learning. There are 4 locations (labeled by different letters) and your job is to pick up the passenger at one location and drop him off in another. You receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions. This environment has 500 different discrete states. This environment is also fully deterministic, not stochastic.

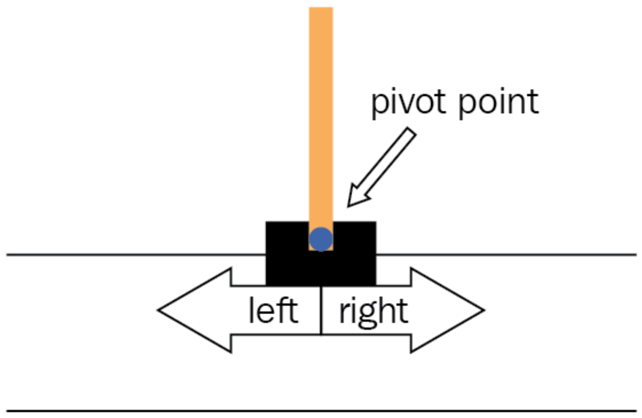
https://gym.openai.com/envs/Taxi-v2/

Action Space:
1. South
2. North
3. East
4. West
5. Pickup
6. Dropoff

Environment returns a vector which is [probability, next state, reward, done]

Cartpole-V0

- A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. Also, this environment has a continuous state space.
- https://gym.openai.com/envs/CartPole-v0/




```
Observation:
    Type: Box(4)
    Num     Observation               Min         Max
    0       Cart Position             -4.8          4.8
    1       Cart Velocity             -Inf          Inf
    2       Pole Angle                 -24°         24°
    3       Pole Velocity At Tip      -Inf          Inf

Action:
    Type: Discrete(2)
    Num     Action
    0       Push cart to the left
    1       Push cart to the right
```

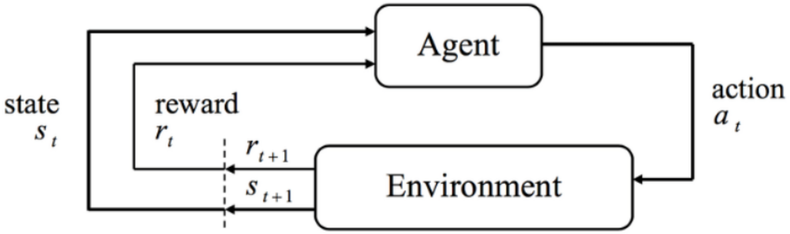
Images is taken from: https://subscription.packtpub.com/book/data/9781789345803/6/ch06lvl1sec47/introducing-cartpole-v1

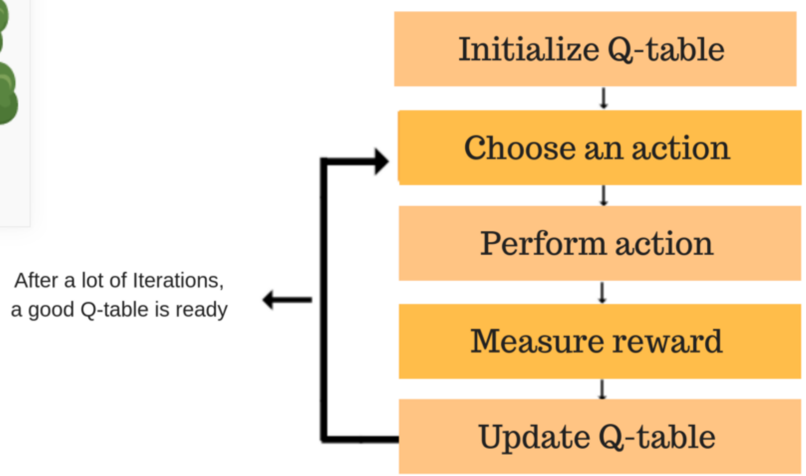
Methods

Q-Learning Algorithm

- Q-Learning algorithm has been developed inspired by Bellman Equation.
- Q-Learning algorithm works with environments which have discrete states and states are defined in the environment.

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$





Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	:	:	:	:	:	:	:
	327	0	0	0	0	0	0
	:	:	:	:	:	:	:
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	:	:	:	:	:	:	:
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	:	:	:	:	:	:	:
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

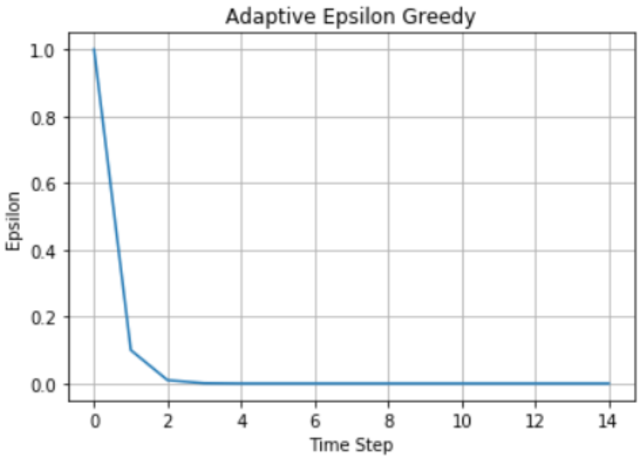
Exploitation and Exploration

Exploration: The agent selects an action in order to explore the environment.
Exploitation: The agent selects an action based on own past experiences.

ε - Greedy Algorithm:
ε should be small like 0.1

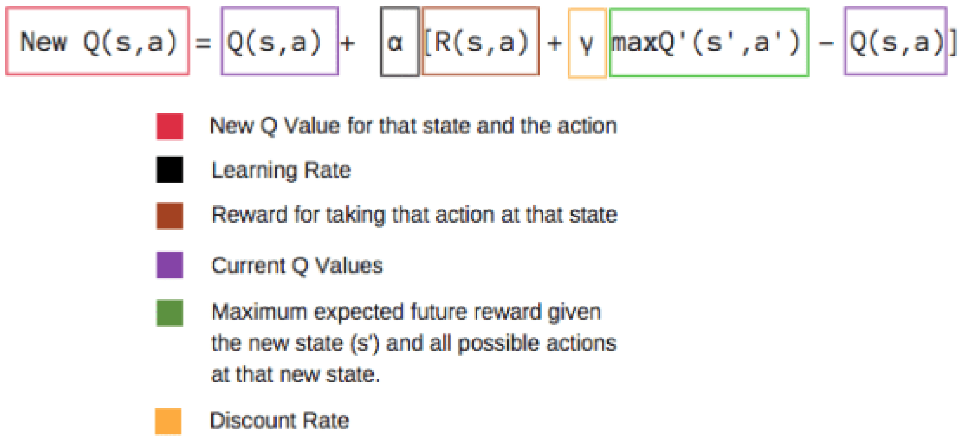
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

Adaptive ε - Greedy Algorithm:
Initializing with 1, decay rate is 0.1
Minimum epsilon is 1e-6



Images is taken from: https://www.ntt-review.jp/archive_html/201811/images/fa1_fig03.jpg

Parameters



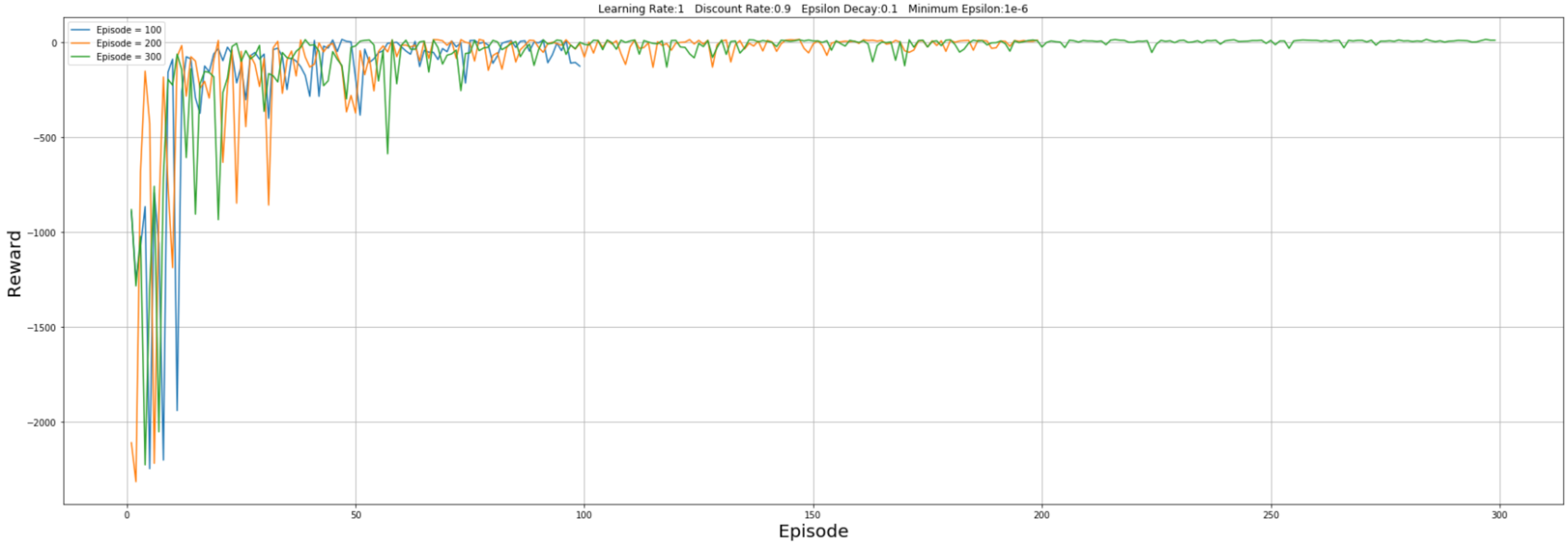
In the Reinforcement Learning, temporal difference is a measure of learning.

Learning Rate:
The learning rate or step size determines to what extent newly acquired information overrides old information. A factor of 0 makes the agent learn nothing, while a factor of 1 makes the agent consider only the most recent information. In fully deterministic environments, a learning rate 1 is optimal. When the problem is stochastic, the algorithm converges under some technical conditions on the learning rate that require it to decrease to zero. In practice, often a constant learning rate is used, such as 0.1

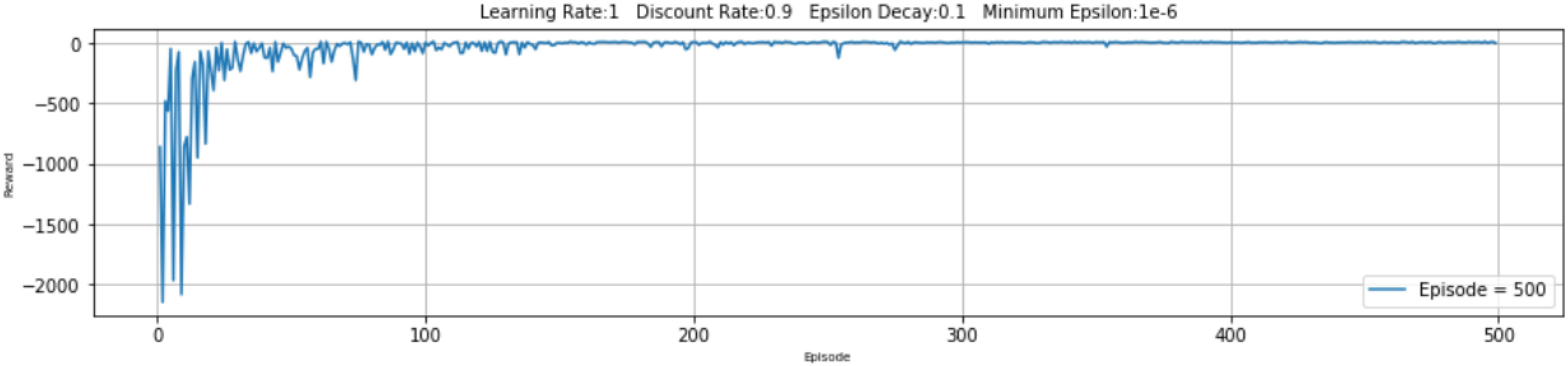
Discount Factor:
The discount factor determines the importance of future rewards. If discount factor is 0, agent only considers the current reward. In case of the high discount factor, agent focuses maximum expected future rewards come from next states.

Experimental Results and Tuning in the Q-Learning Algorithm

Episode



Learning Rate:1 Discount Rate:0.9 Epsilon Decay:0.1 Minimum Epsilon:1e-6
Episode = 100
Episode = 200
Episode = 300
Reward
Episode
0
−500
−1000
−1500
−2000
0
50
100
150
200
250
300

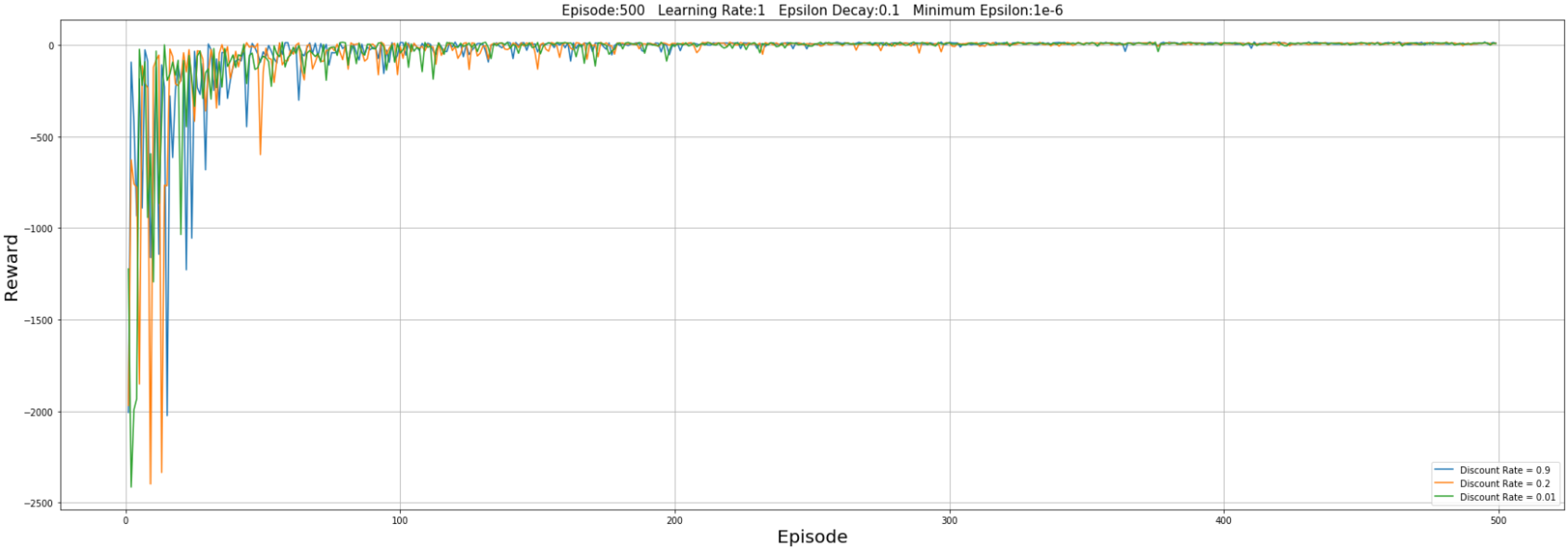


Learning Rate:1 Discount Rate:0.9 Epsilon Decay:0.1 Minimum Epsilon:1e-6
Episode = 500
Reward
Episode
0
−500
−1000
−1500
−2000
0
100
200
300
400
500

Learning Rate

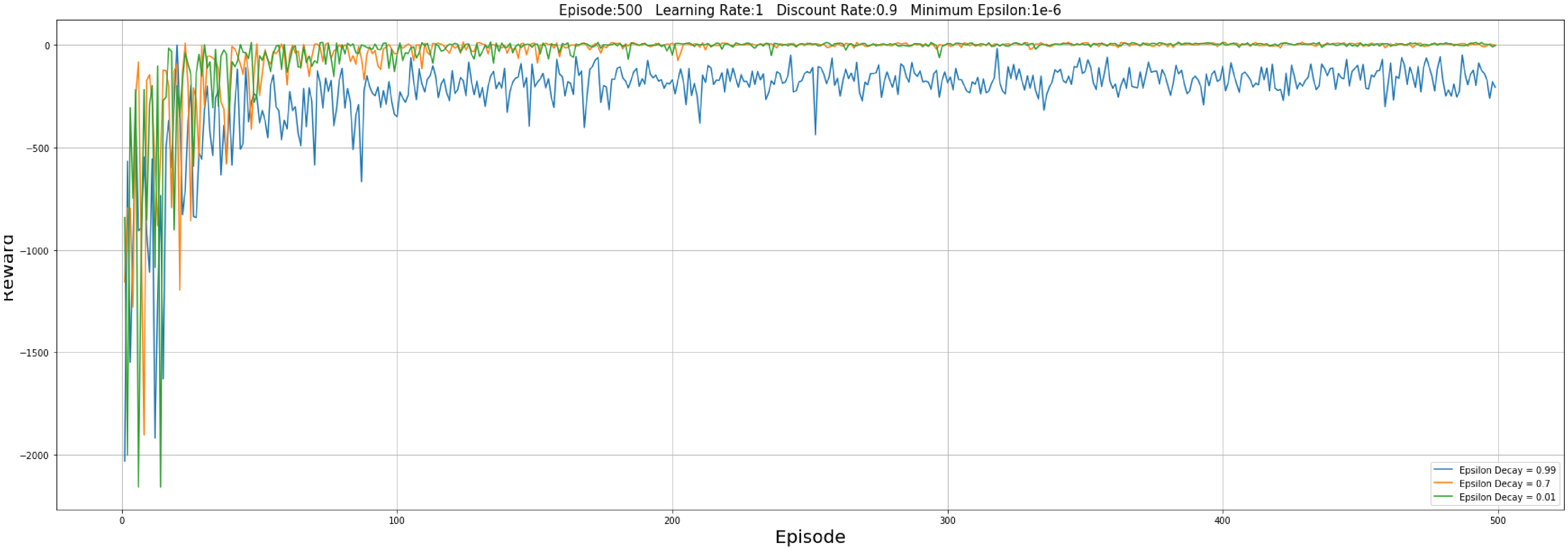
Discount Rate

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \bigg)}^{\text{temporal difference}}$$



The results are similar because the expected future reward is not valuable enough. Focusing on the current reward is enough to reward convergence.

Epsilon Decay

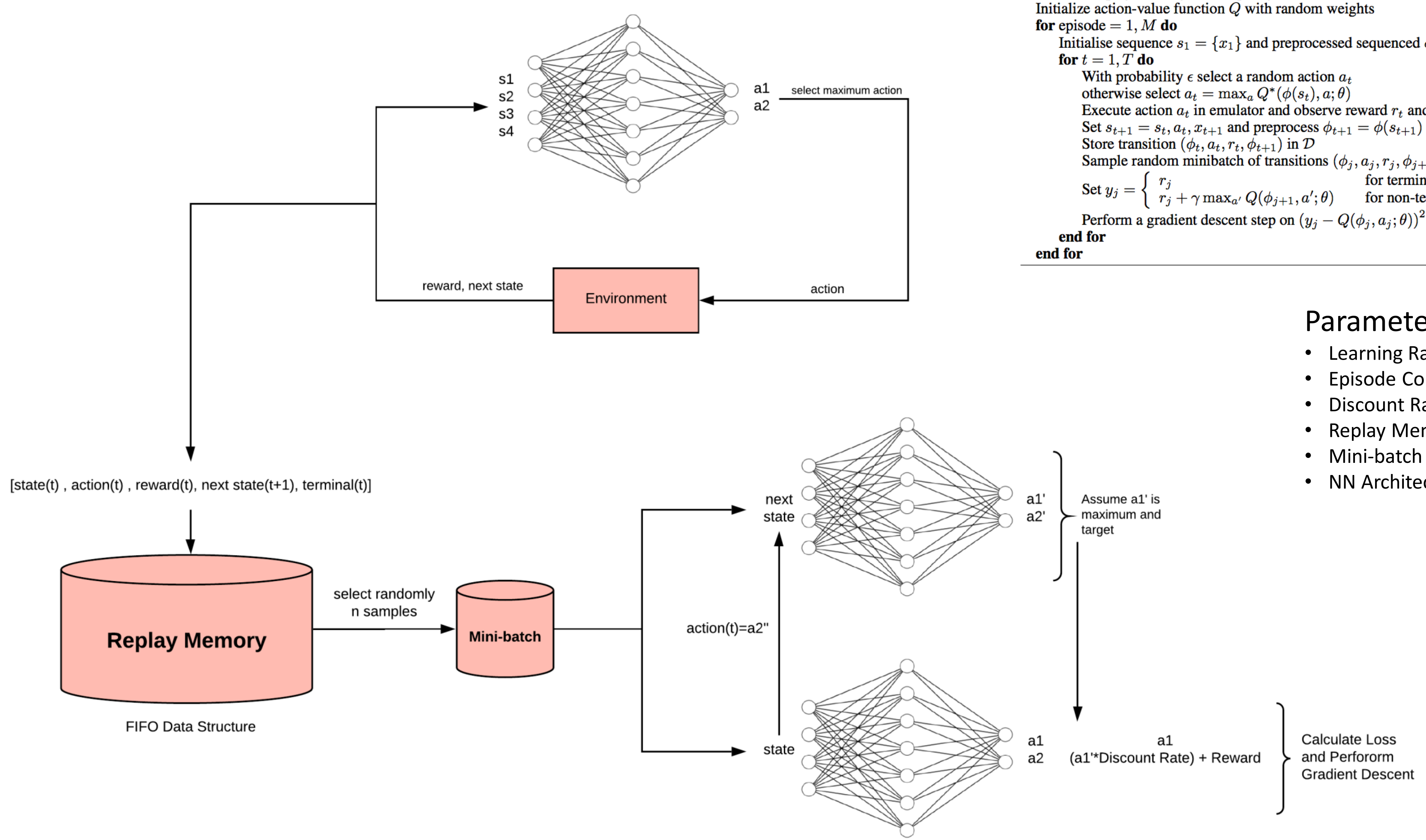


If the epsilon is high, the agent selects action randomly.

Deep Q-Learning Algorithm

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

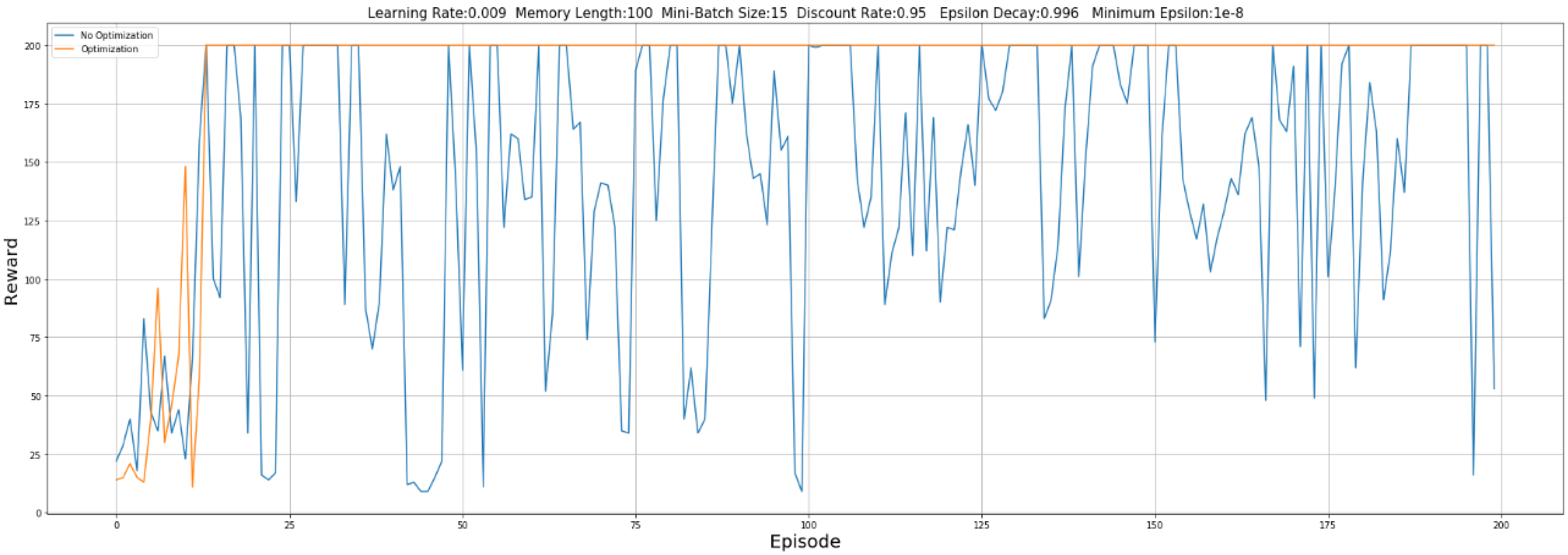


Parameters:

- Learning Rate
- Episode Count
- Discount Rate
- Replay Memory Size
- Mini-batch Size
- NN Architecture

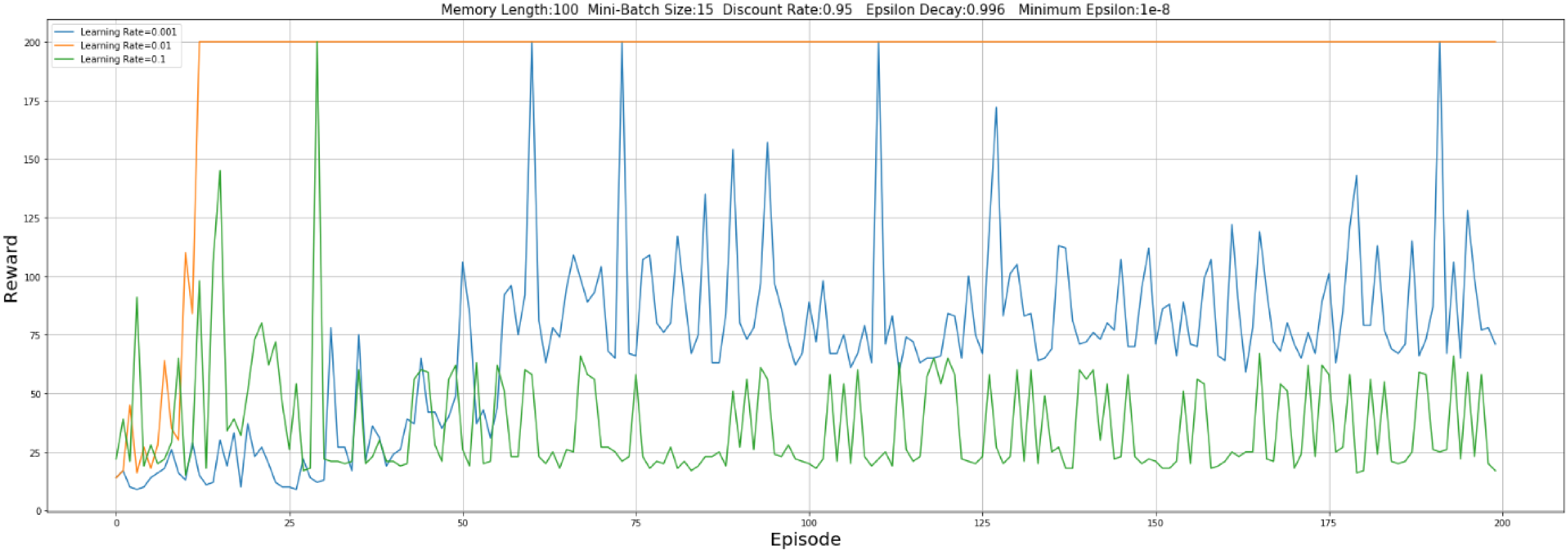
Experimental Results and Tuning in the Deep Q-Learning Algorithm

Stopping of Training

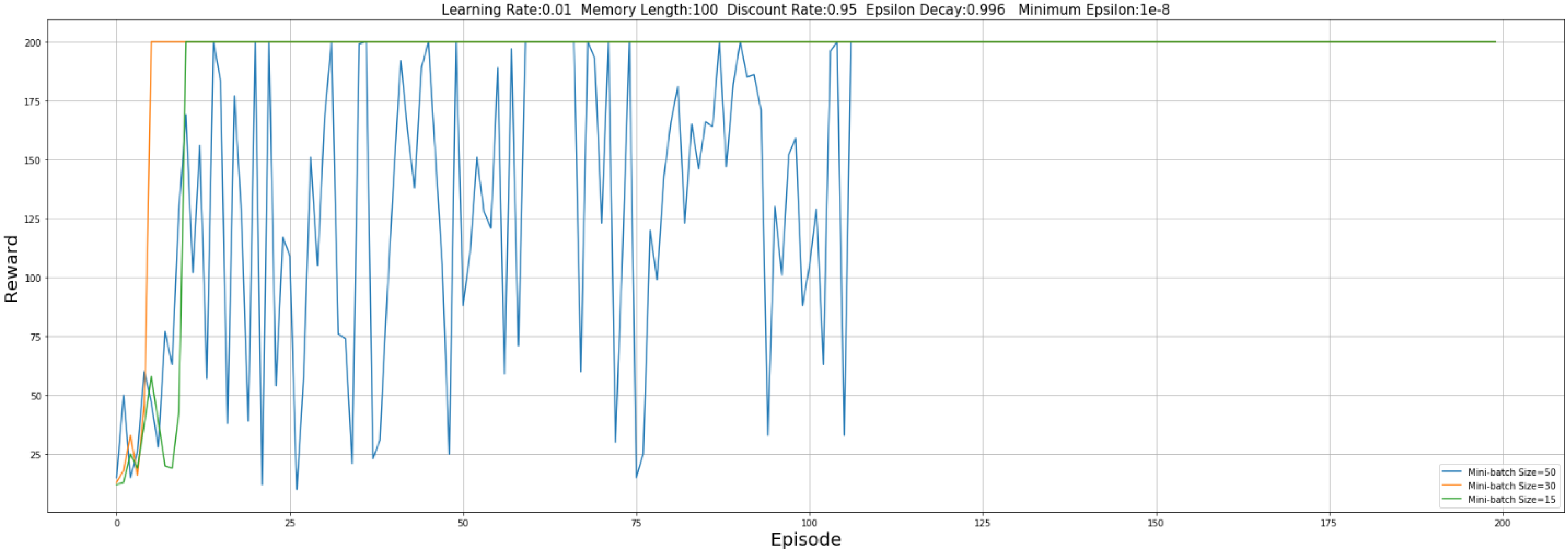


- If the reward is maximized at the last time step, stop backward propagation of Neural Network, and epsilon is equal to zero.
- If the reward is not maximized at the last time step, pursue the backward propagation.
- This optimization was applied to the following experimental setups.

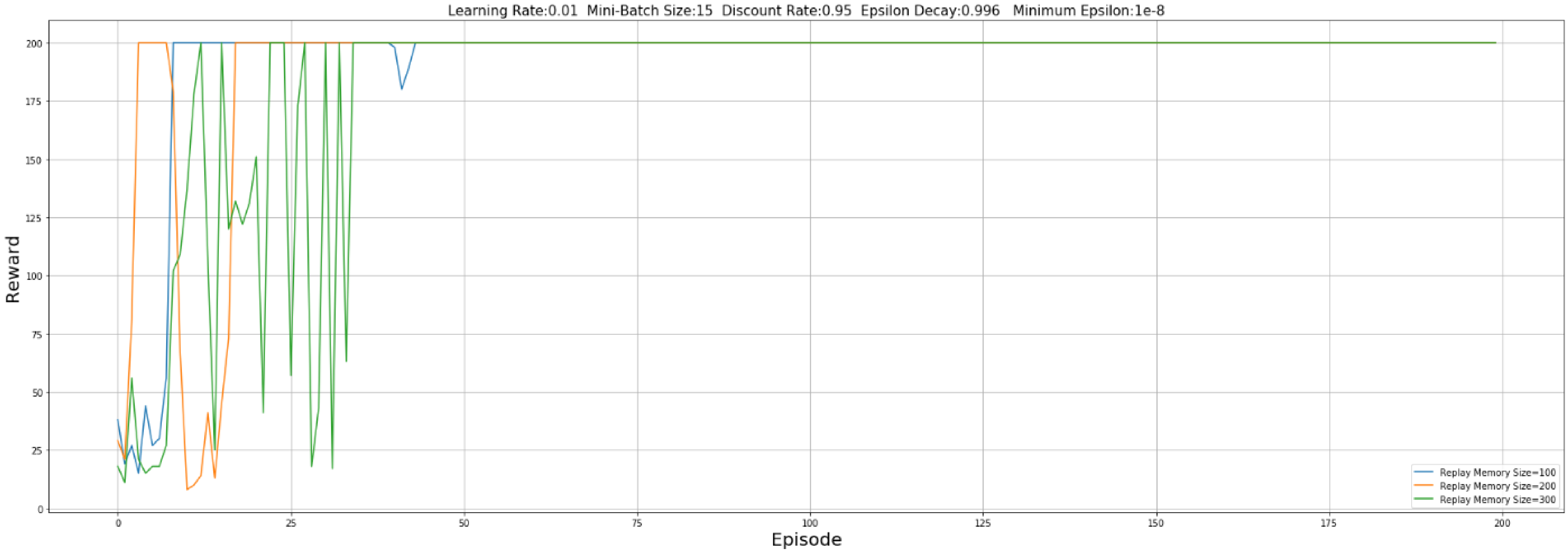
Learning Rate



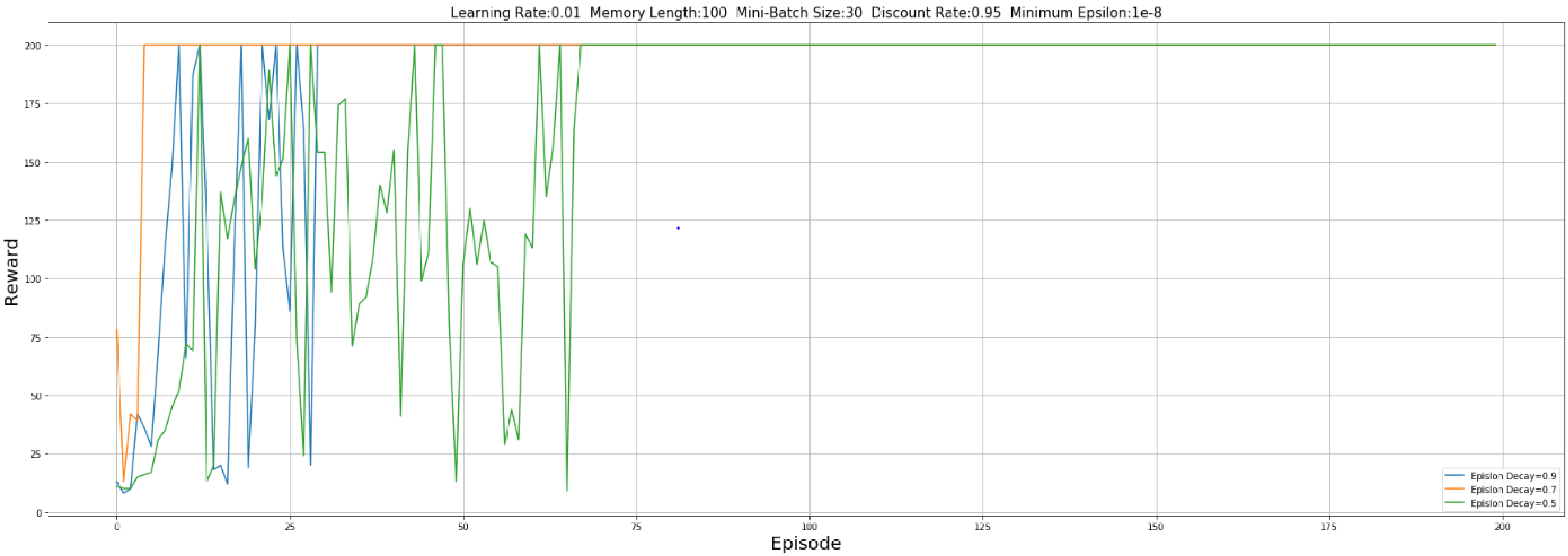
Mini-Batch Size



Replay Memory Size

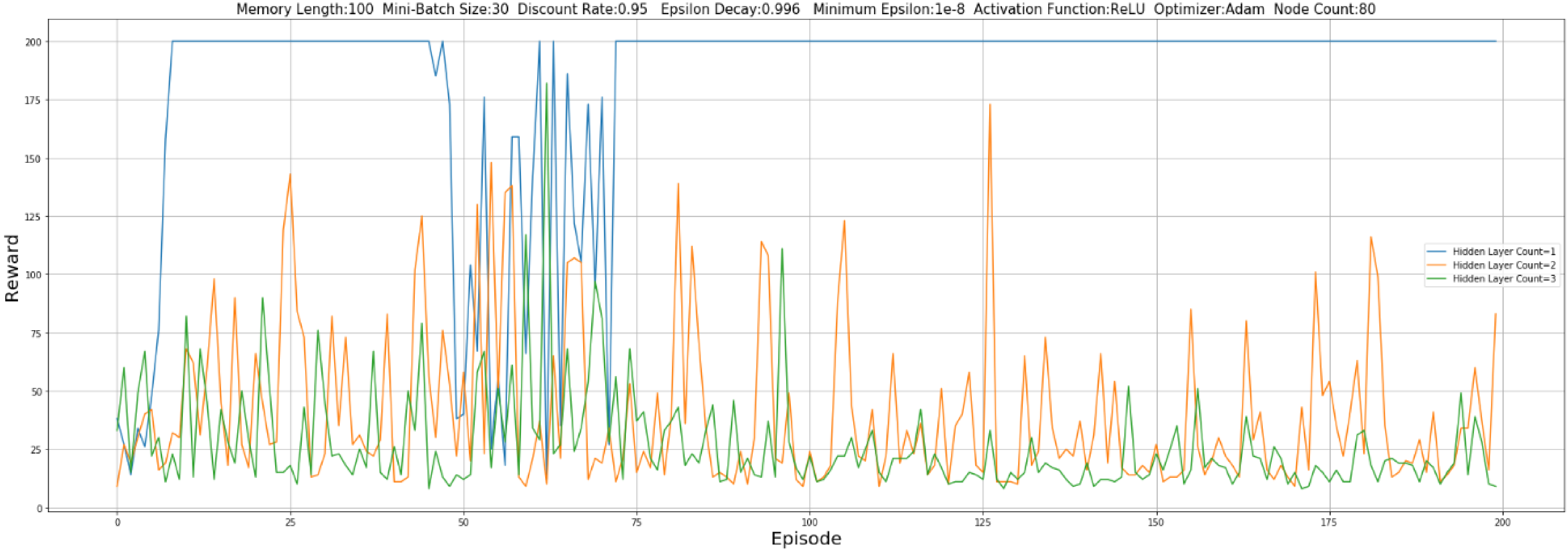


Epsilon Decay



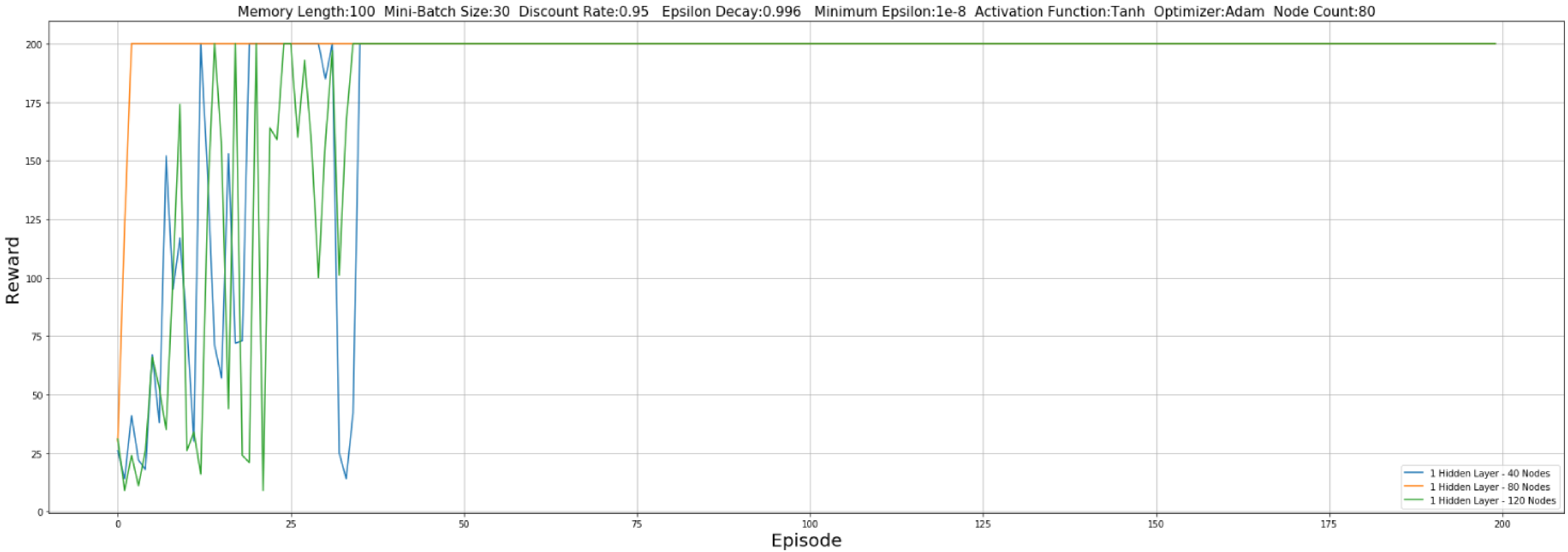
- It indicates that there must be a balance between exploration and exploitation.

Neural Network Architecture:
Hidden Layer Count



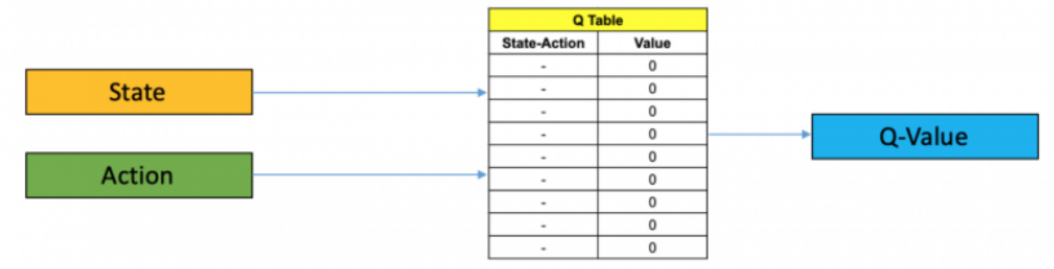
Neural Network Architecture:

Hidden Layer Size

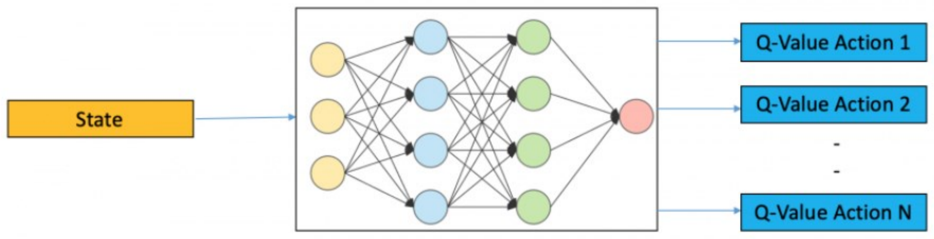


Conclusion

- **Best Tuning Parameters:**
  - **Q-Learning:** Episode:500, Learning Rate:1, Epsilon Decay:0.1, Discount Rate:0.9
  - **Deep Q-Learning:** 1 Hidden Layer, 80 nodes, Memory Length:100, Mini-batch Size:30, Discount Rate:0.95, Epsilon Decay:0.996, Activation Function: Tanh, Optimizer:Adam, Learning Rate: 0.01, Episode:500



Q Learning



Deep Q Learning

- The Q-Learning algorithm works well with the environment that has small discrete sate space.
- The Deep Q-Learning algorithm works well with the environment that has continuous state space. Also, it can handle a large amount of states according to Neural Network architecture.

Images is taken from: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

References

- Intellipaat.com. 2020. What Is Agent In Reinforcement Learning? - Intellipaat Community. [online] Available at: <https://intellipaat.com/community/44608/what-is-agent-in-reinforcement-learning> [Accessed 26 May 2020].
- Osiński, B. and Budek, K., 2020. What Is Reinforcement Learning? The Complete Guide - Deepsense.Ai. [online] deepsense.ai. Available at: <https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/> [Accessed 26 May 2020].
- KDnuggets. 2020. Three Things To Know About Reinforcement Learning - Kdnuggets. [online] Available at: <https://www.kdnuggets.com/2019/10/mathworks-reinforcement-learning.html> [Accessed 26 May 2020].
- Image: https://www.ntt-review.jp/archive_html/201811/images/fa1_fig03.jpg
- Medium. 2020. Deep Q Learning Explained. [online] Available at: <https://medium.com/@uwdlms/deep-q-learning-explained-2bd591d03f25> [Accessed 26 May 2020].
- Mnih, V., 2020. [online] Cs.toronto.edu. Available at: <https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf> [Accessed 26 May 2020].
- En.wikipedia.org. 2020. Q-Learning. [online] Available at: <https://en.wikipedia.org/wiki/Q-learning> [Accessed 26 May 2020].
- Team, D., 2020. Python Ile Yapay Zeka: A'dan Z'ye Reinforcement Learning (7). [online] Udemy. Available at: <https://www.udemy.com/course/python-ile-yapay-zeka-adan-zye-reinforcement-learning/> [Accessed 26 May 2020].