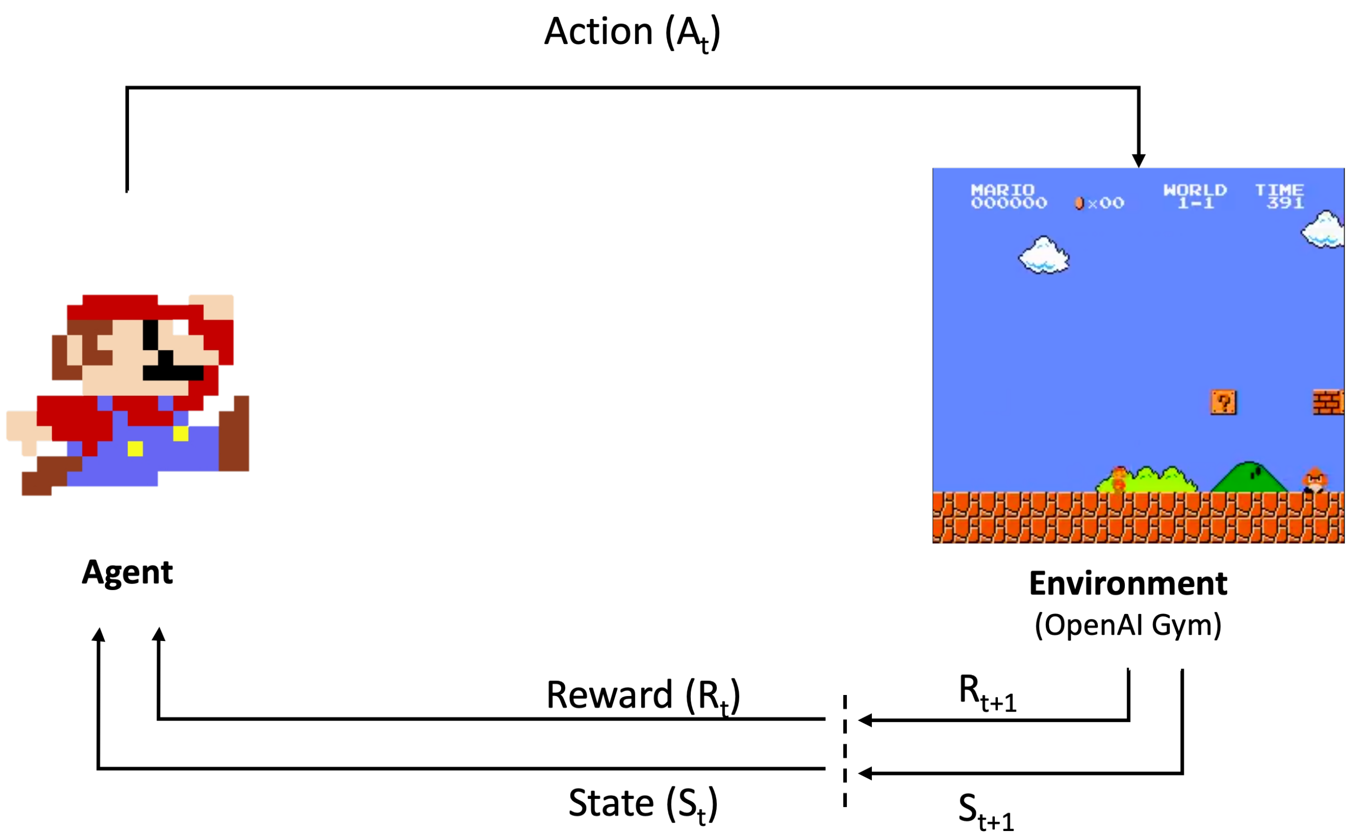
Group 20!

CM50270 – Reinforcement Learning

Final Project Presentation

Problem Description

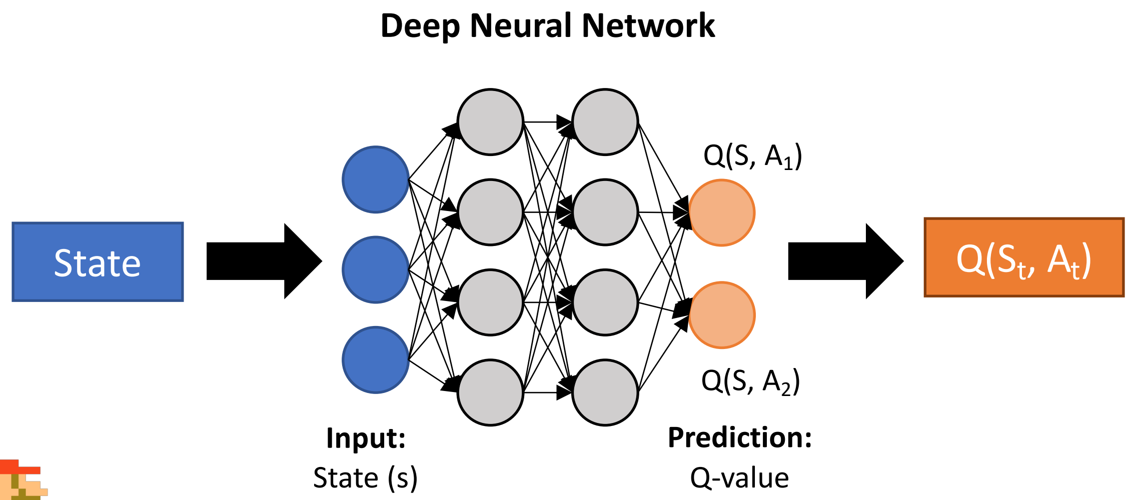
- Train RL agents to play **Super Mario Bros** World 1 Stage 1



RL Algorithms Suitable for the Problem

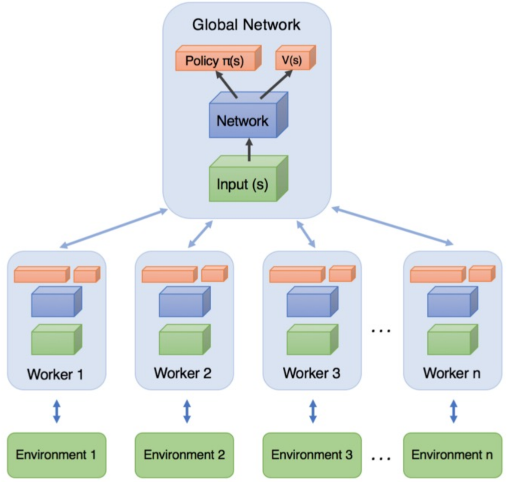
Deep Q-Network

- Value-based learning
- Instead of storing Q-values in tabular form, uses a DNN for prediction of Q-values
- Uses Experience Replay to learn from past experiences
- Method used in official PyTorch tutorial (clears the first stage, suggested running at least 40,000 episodes)



Actor Critic

- Combines both value-based and policy gradient learning
- Composes of 2 parts:
  - **Actor**: Estimates actions based on policy, updates policy according to critic's feedback
  - **Critic**: Estimates values of actions taken and evaluate the action
- In a GitHub repository, 19 stages solved using A3C method



(Learn Reinforcement Learning (4) - Actor-Critic, A2C, A3C · greentec's blog, 2022) [Link]

Implementation Process

Agents Development → Hyperparameter Optimisation → Further Optimisation

Agents Development

- DQN
- Double-DQN
- Dueling-DQN

Hyperparameter Optimisation

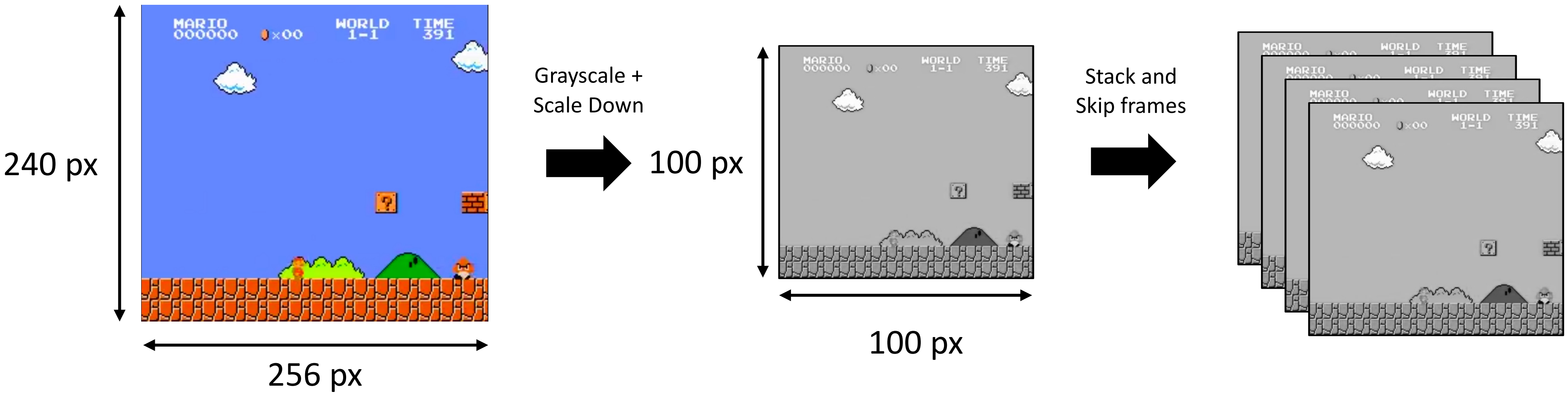
- Gamma
- Epsilon decay rate
- Target model update rate for Double-DQN

Further Optimisation

- Prioritised Experience Replay

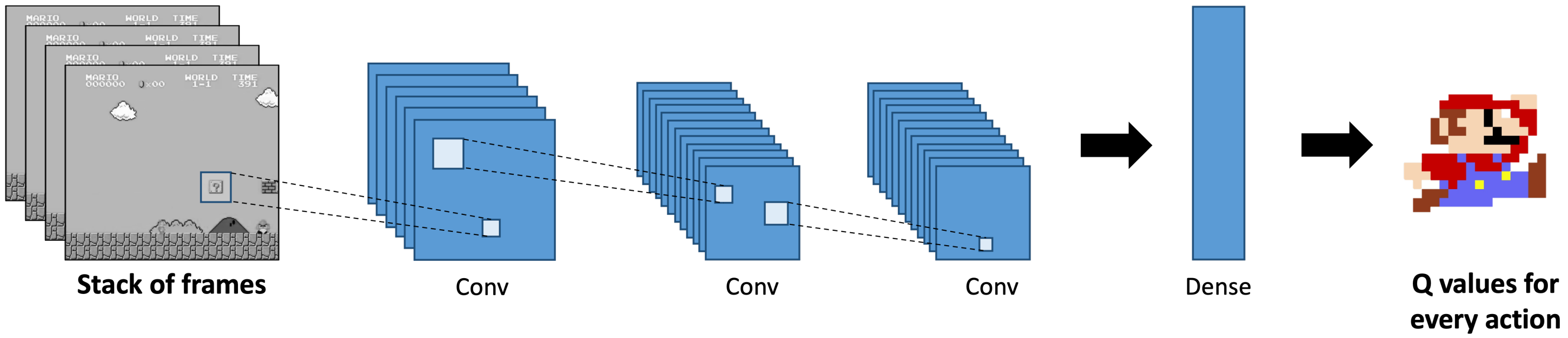
Frames Preprocessing

- States represented as pixel frames
- Referenced DeepMind Atari agents' environment preprocessing steps to downsize the number of feature vectors in the DNN → enhances computational efficiency



Agents Implemented - DQN

- Use CNN as the DQN for computing the Q-values
- Architecture of the CNN:



- At each timestep, perform **Experience Replay** to update the model weights:

$$y_i = \begin{cases} r_i & \textit{if terminal } s_{i+1} \\ r_i + \gamma \max_{a'} \hat{Q}(s_{i+1}, a', \theta) & \textit{otherwise} \end{cases}$$

Rewards: clipped at [-15, 15]

Gradient descent step: $\nabla_\theta L_\delta(y_i - \hat{Q}(s_i, a_i, \theta))$

Huber Loss

Agents Implemented – Double-DQN

- Problem of DQN: **Shifting target**
- Solution: Introduction of a **target network**
- At every time step:
  - **Choose best action** according to **online network (Q_1)** prediction
  - Perform Experience Replay to update weights of online model based on Q-value predictions of target model **(Q_2)**

$$y_i = \begin{cases} \boxed{r_i} & \textit{if terminal } s_{i+1} \\ \boxed{r_i} + \gamma \max_{a'} \widehat{Q_2}(s_{i+1}, a', \theta_2) & \textit{otherwise} \end{cases}$$

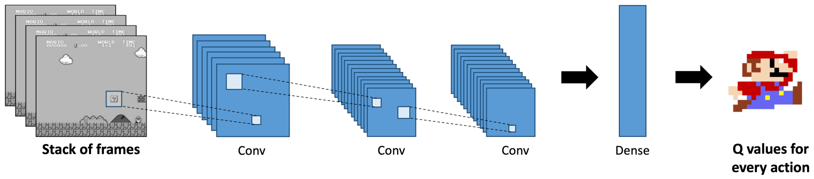
Rewards: clipped at [-15, 15]

Gradient descent step: $\nabla_\theta \boxed{L_\delta}(y_i - \widehat{Q_1}(s_i, a_i, \theta))$

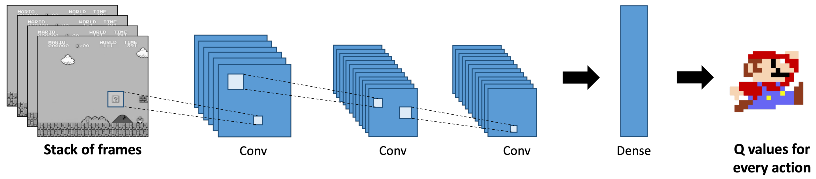
Huber Loss

- Target network copies the weights of online network at every N steps

Online network



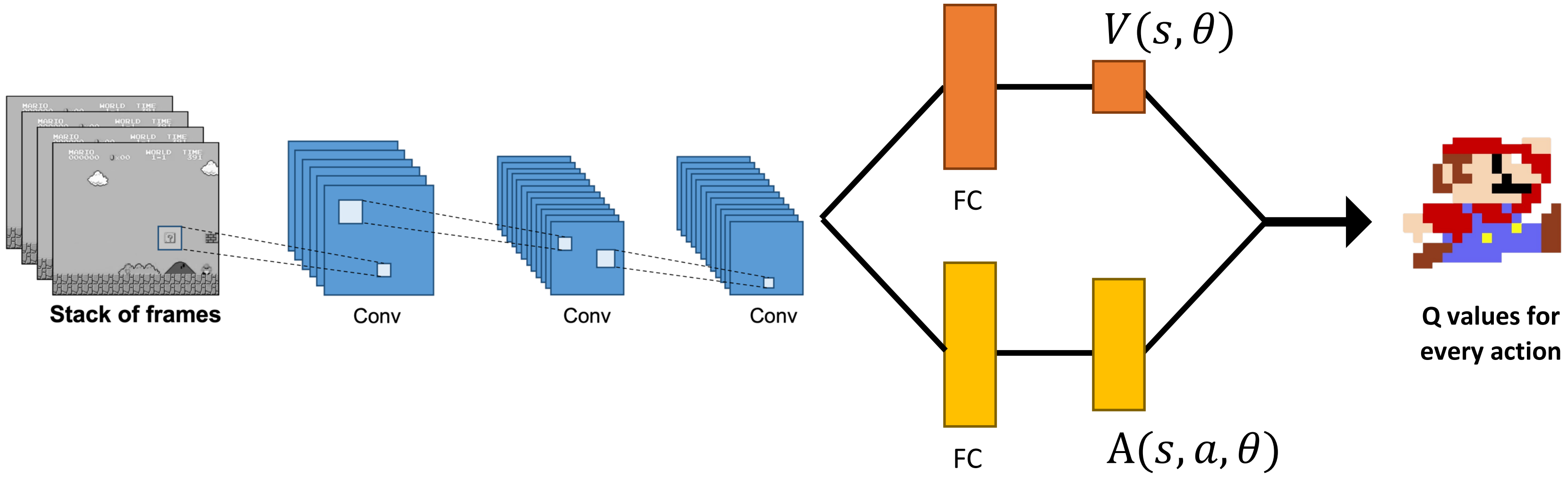
Target network (Frozen weights)



copy weights from **online** every N steps

Agents Implemented – Dueling-DQN

- Introduces a **value** branch and an **advantage** branch in the CNN

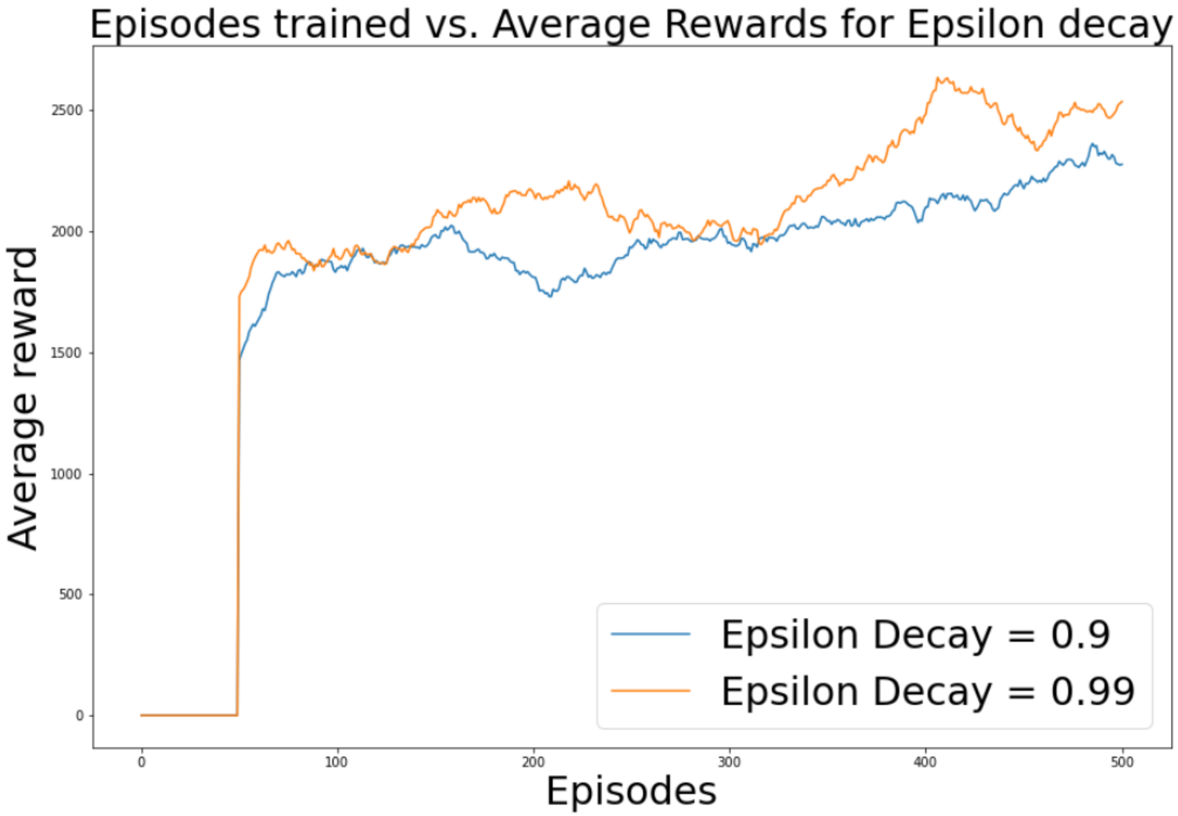


Final aggregation of streams:

$$Q(s,a,\theta) = V(s,\theta) + A(s,a,\theta) - \frac{1}{|A|}A(s,a,\theta)$$

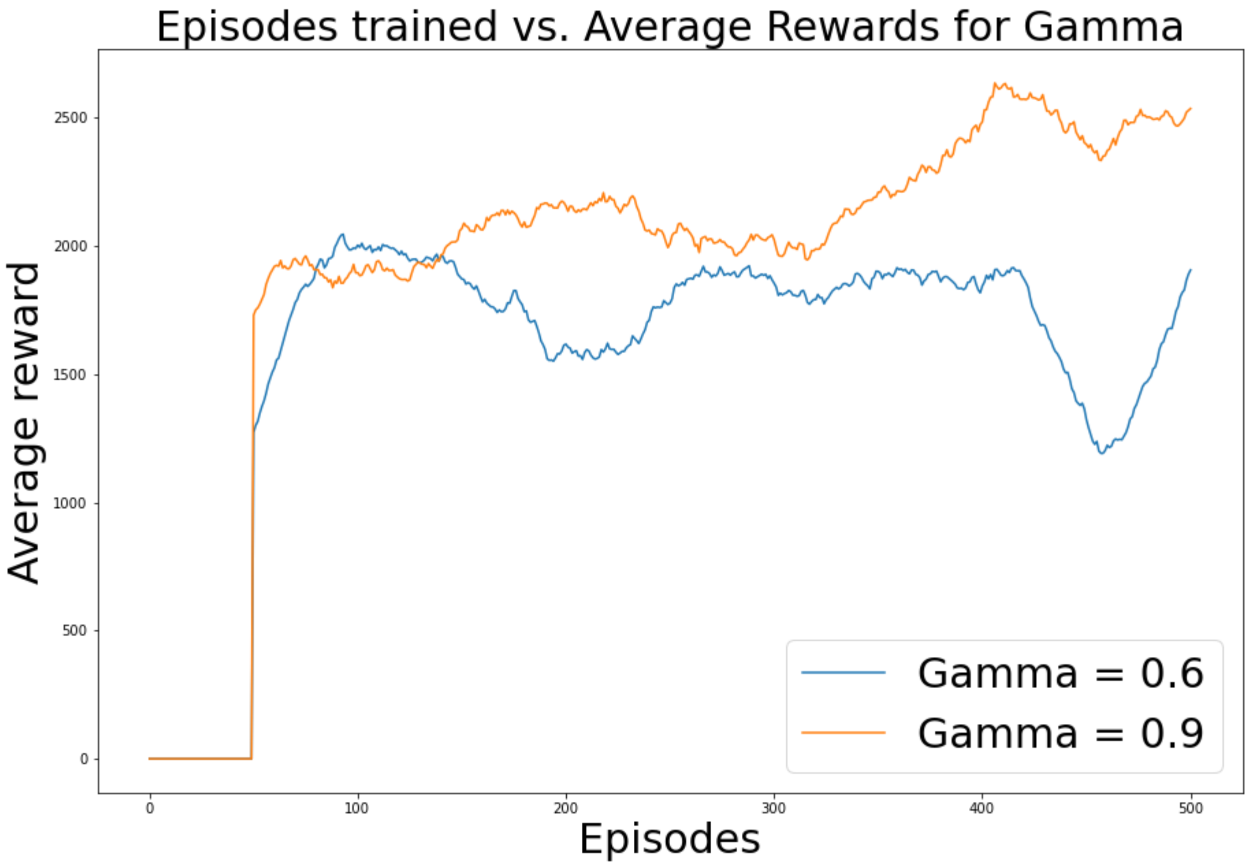
Hyperparameter Optimisation

- **Epsilon Decay-** Controlled the ratio of time the agent spent exploring vs exploiting



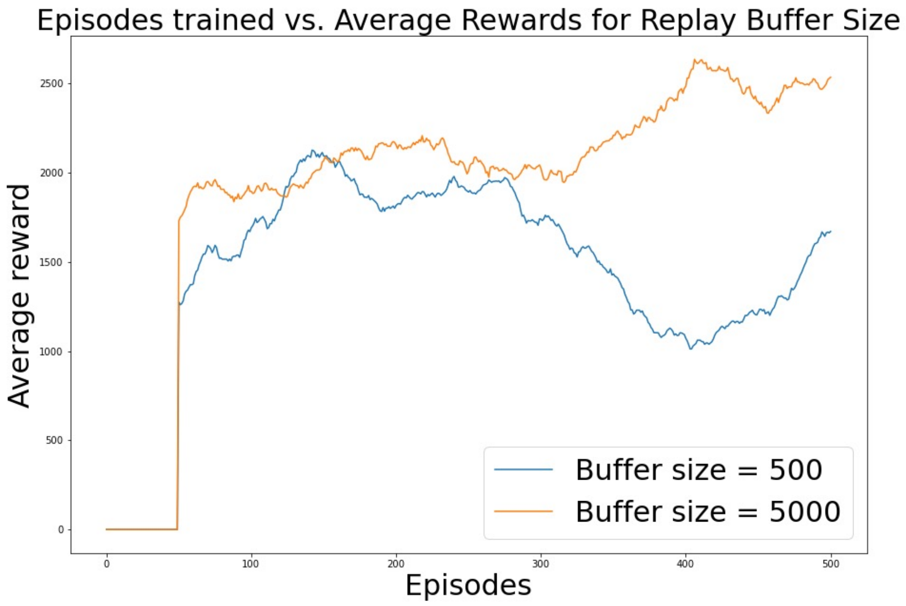
Hyperparameter Optimisation

- **Gamma-** Decides how much importance is given to future rewards

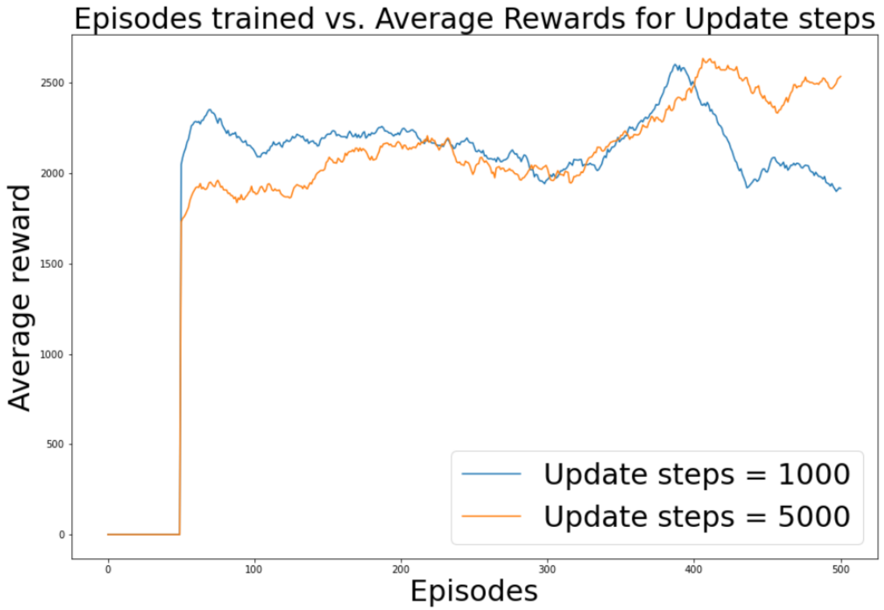


Hyperparameter Optimisation

- **Replay Buffer Size-** How many tuples of experience we retain



- **Update steps-** How often we sync the weights from the DQN with the target network

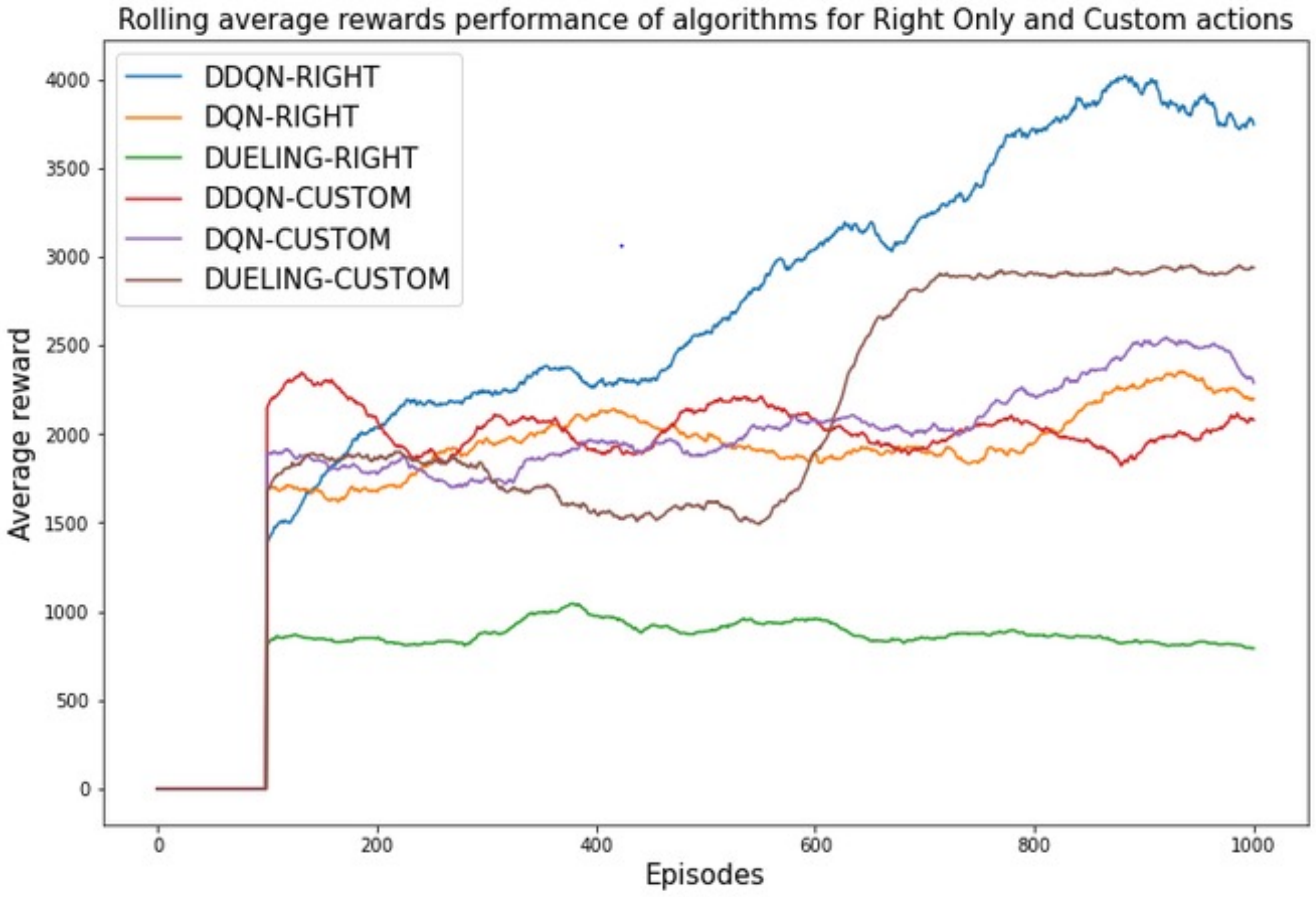


Performance Comparison

Agent types:
DQN
DDQN
Dueling-DQN

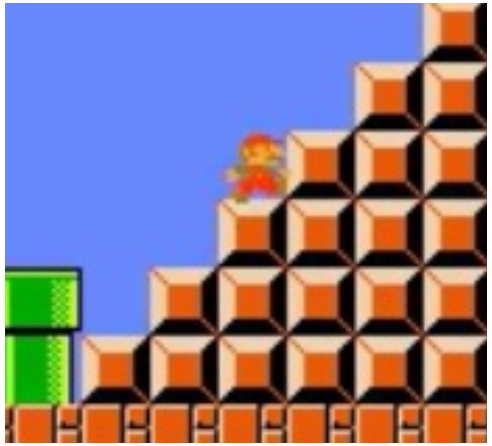
Trained on: 1000 episodes

Actions:
Right only
Custom: Right and Right/A

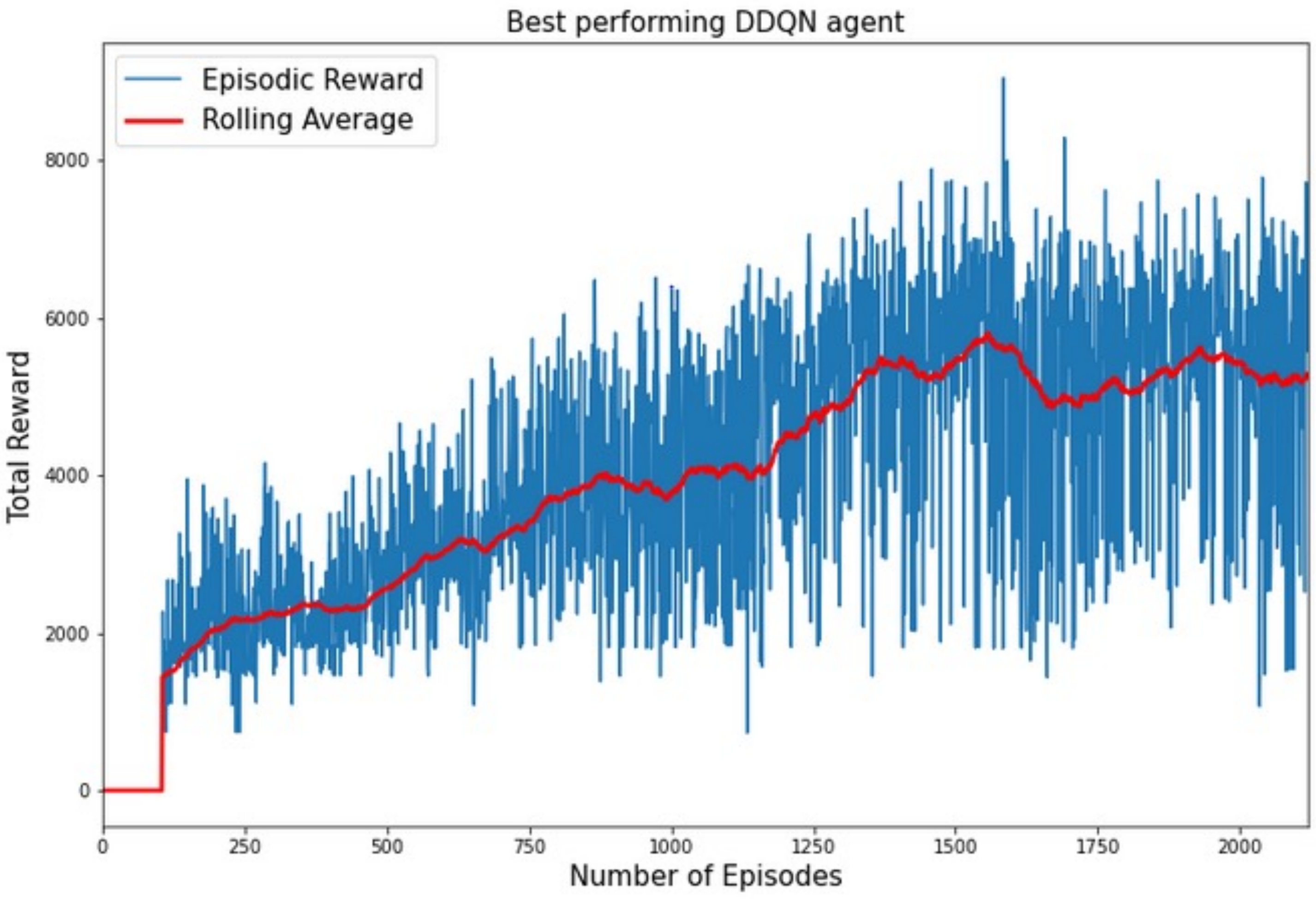


Performance of our Best Agent

```
Model type: Double DQN (DDQN)
Experience replay: Uniform
Mario's moveset: RIGHT_ONLY (no op, right, right + A, right + B, right + A + B)
Max memory for experiences: 20000
Max epsilon: 1.0
Min epsilon: 0.1
Epsilon decay: 0.99
Steps until target network updates: 5000
Batch size for experience replay: 64
Gamma: 0.9
Learning rate: 0.00025
Episodes of training: 2500
Environment: SuperMarioBros-v0
Record rate: 20 episodes
```




Learning Curve



THANK YOU