# Simulating Game Agent using Q-Network (Reinforcement Learning Technique)

Project Group P25

- Advait Trivedi (astrived)
- Sainag Shetty (sgshetty)
- Pratik Kumar Jain (pjain22)
- Aaroh Gala (agala)

# Problem Statement

- Training an agent to learn to play Flappy Bird using Reinforcement Learning Techniques.
- Game:
  - The objective was to direct a flying bird, who moves continuously to the right, between sets of Mario-like pipes.
  - It can perform either of the 2 actions; Flap or Not Flap.
  - While not flapping, the bird falls due to gravity.



# What is Reinforcement Learning

- RL is learning how to map states to actions, so as to maximize a numerical reward over time
- After performing action a in state s, the environment assumes the new state, and the agent gets a reward. (S-A-R)
- An RL agent must learn by trial-and-error.



# What is Q-Learning?

- In Q-learning, an agent tries to learn the optimal policy from its history of interaction with the environment
- A history of an agent is a sequence of state-action-rewards:

$$\langle s0,a0,r1,s1,a1,r2,s2,a2,r3,s3,a3,r4,s4\ldots\rangle$$

- Q-Function:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left( R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right)$$

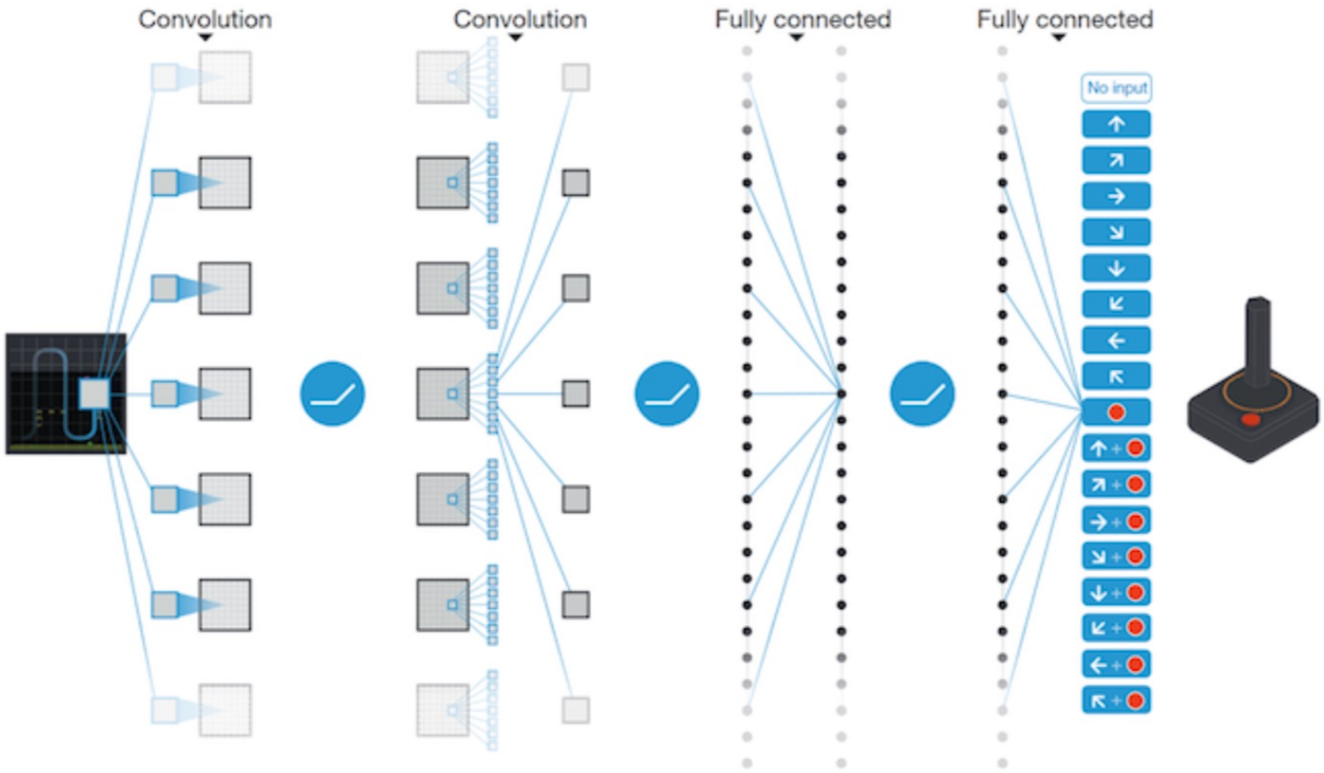


# Deep Q-Model

- The initial convolution layers handle image feature extraction.
- The subsequent convolution layers give a more human like input representation of the game environment.
- The final hidden layer is fully-connected consisted of 512 rectifier units.
- The output layer is a fully-connected linear layer with a single output for each valid action.

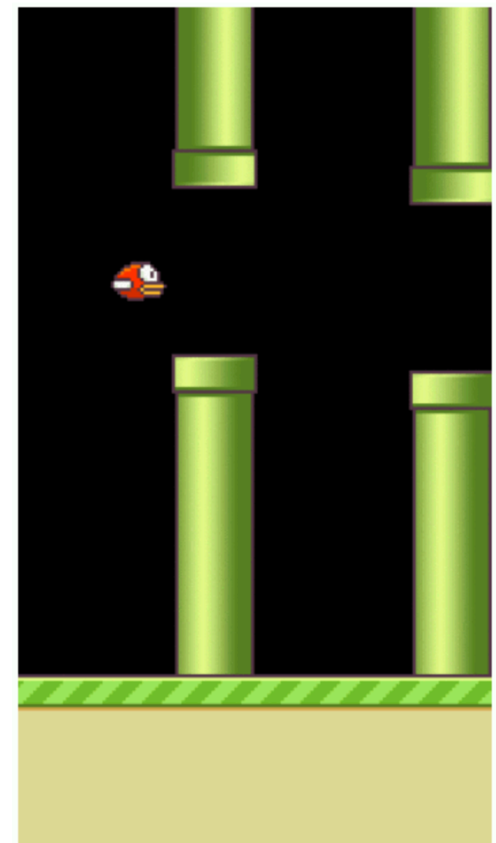
| Layer number | Property of layer | Activation Function |
|---|---|---|
| 1st convoluted layer | 32 filters of 8 x 8 with stride 4 | Relu |
| 2nd Convoluted layer | 64 filters of 4 x 4 with stride 2 | Relu |
| 3rd Convoluted layer | 64 filters of 3 x 3 with stride 1 | Relu |
| Fully connected hidden layer | 512 rectifier units | Relu |
| Output layer | Output unit: Flap/No Flap decision | - |

# Model Architecture



# Experimental Setup

- Environment:
  - State:
    - Current Frame of the Game
  - Action:
    - Flap(ACTION = 1)
    - No Flap(ACTION = 0)
  - Reward:
    - Agent stays alive (REWARD = +0.1)
    - Agent passes through the tunnel (REWARD = +1)
    - Agent dies ( REWARD = -1)



# Elements

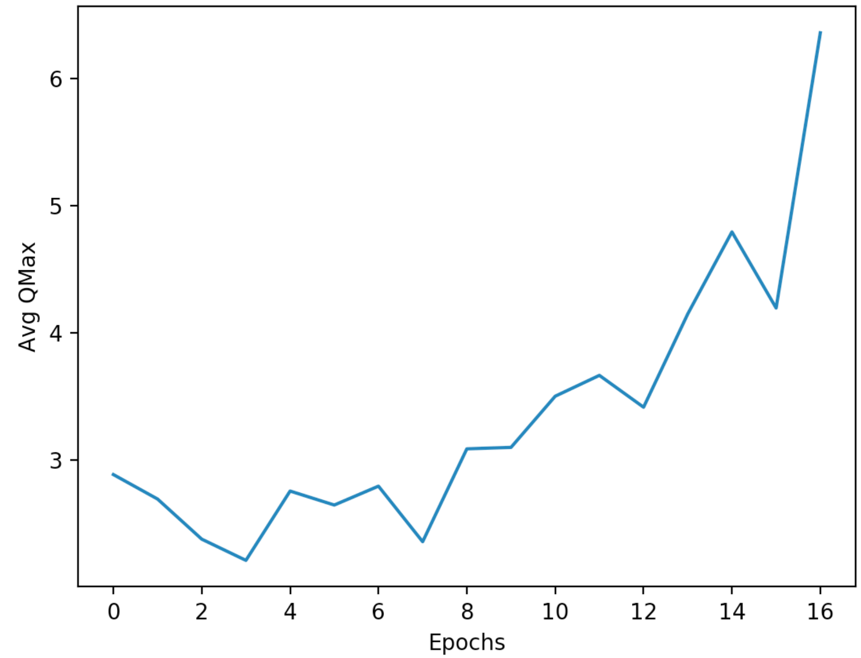
- Mean squared error (MSE), or the **loss function** is given as:

$$L = [r + \gamma max_{a'} Q(s^{'}, a^{'}) - Q(s, a)]^2$$

- **Adaptive Moment Estimation** (Adam) is employed as the optimization algorithm.
- An **Epsilon** greedy approach is an approach in which the policy incorporates exploring a random action some percentage of the time.(Exploration – Expliotation)

# Results

- Figure on the right shows the plot of the Average Q-Max value vs. No. of Epochs
- It shows that the average predicted Q increases with the increase in the number of Epochs
- One Epoch corresponds to 10000 Timestamps.
- This suggests that the method is able to train large neural networks using Reinforcement Learning.



# Conclusion

- After training the agent for 15 epochs(of 10000 Timestamps each), the agent learned to survive for a longer period.
- Demo.

# Acknowledgements


- [1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

- [2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), pp.529-533.

- [3] Sutton, R.S. and Barto, A.G., 1998. Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

- [4] Ponce, H., & Padilla, R. (2014, November). A hierarchical reinforcement learning based artificial intelli- gence for non-player characters in video games. In Mexican International Conference on Artificial Intelligence (pp. 172-183). Springer, Cham.

- [5] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).