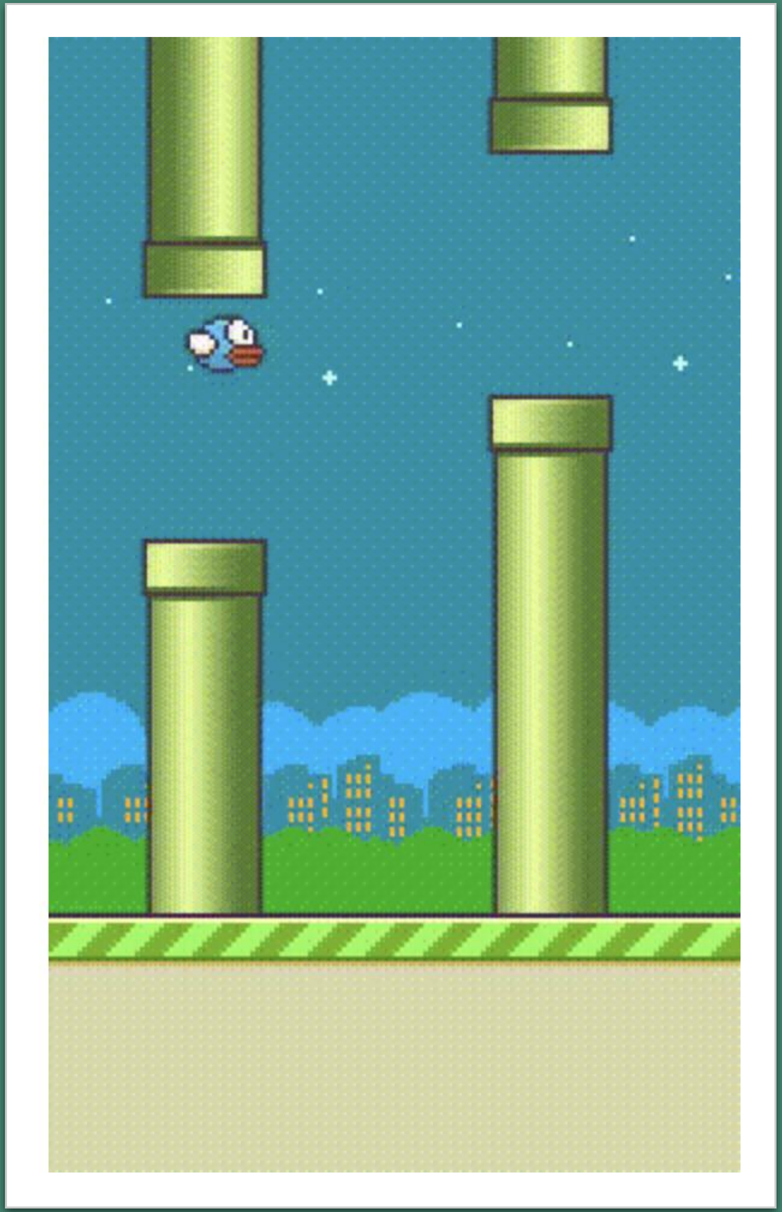


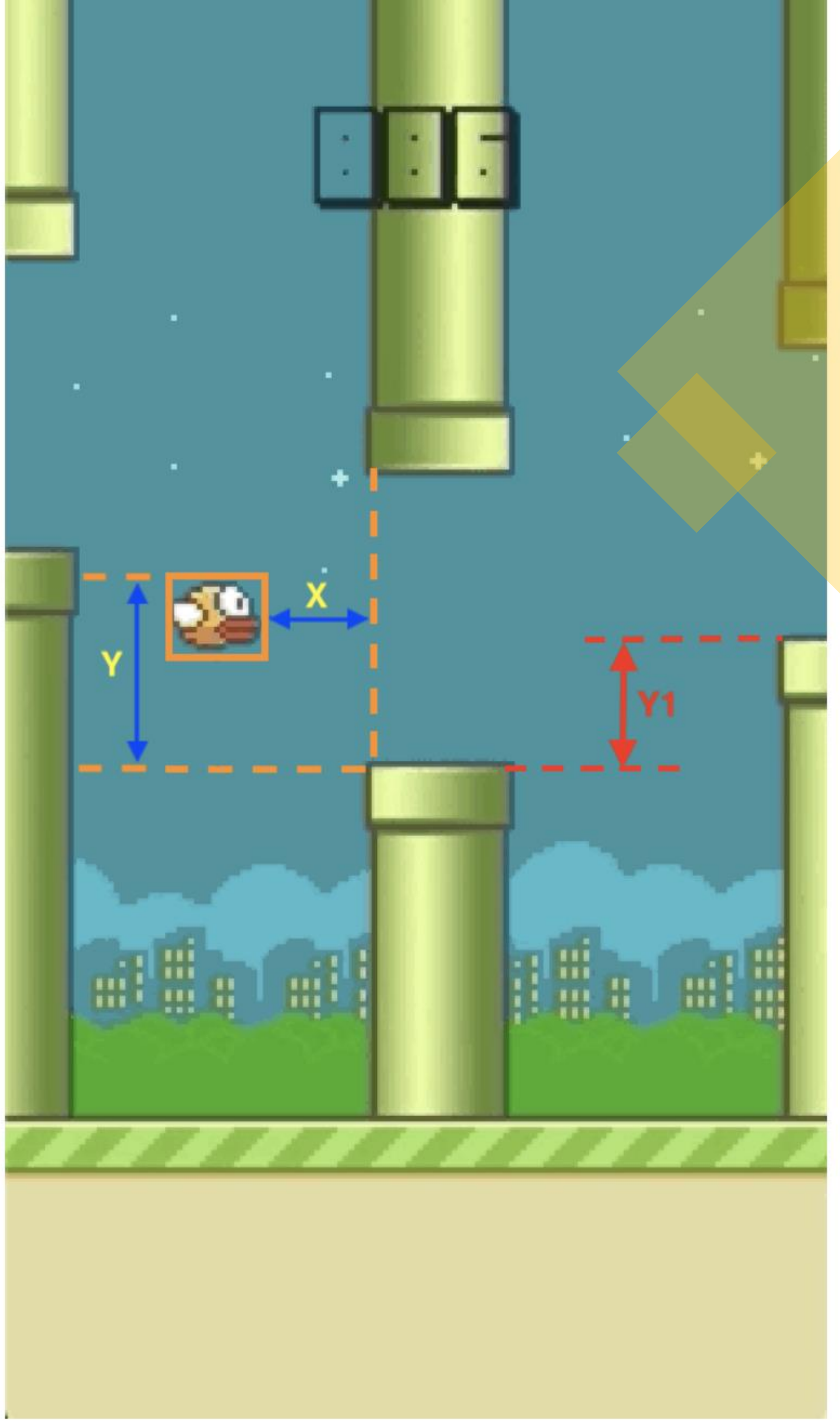
Playing Flappy Bird with Deep Reinforcement Learning

Team members

- Pranav Lodha
- Subarna Chowdhury Soma
- Jeyasri Subramanian

RL Terminology

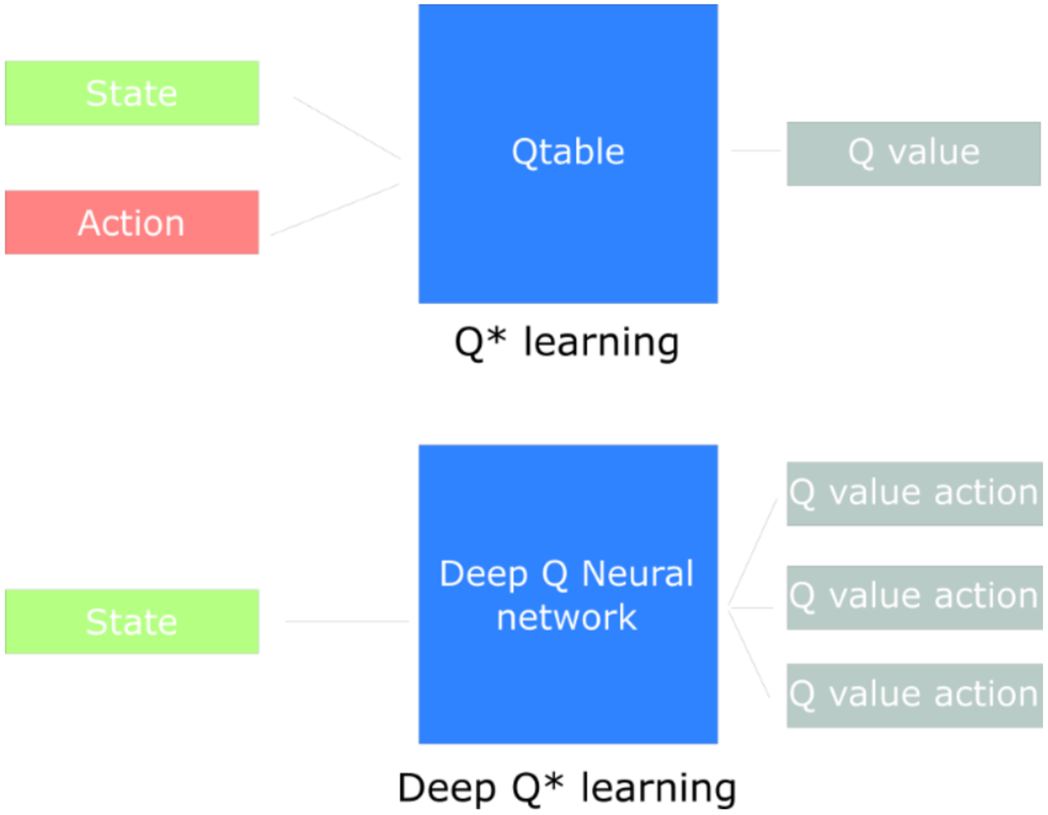
- Agent
  - Flappy bird
- Environment
  - Uneven Pipe in an infinite trial
  - 288 X 512 X 3 RGB image
- Observation
  - [bird height, bird speed, distance to next pipe(x), height of next pipe(y), distance to second pipe, height of second pipe(y1)]
- Reward
  - +1 for positive reward (escaping death)
- Action
  - Flap or do nothing
- Terminal state
  - If the bird makes contact with the ground, pipes or goes above the top of the screen the game is over.



Problem Statement

- Train Flappy bird agent to play continuous using Deep Reinforcement Learning algorithm
- Apply Double Dueling Deep Q Networks(D3QN) with Prioritized Experience replay
- Fine-tune hyperparameters to improve performance
- Compare with other reinforcement algorithm

Deep Q-Network (DQN)



- Q Learning is off policy learning method in reinforcement learning which is a development over on-policy Temporal Difference control algorithm
- Deep Q learning is based on Deep Neural Network
- It takes current state in the form of image or say continuous value and approximates Q-values for each action based on that state.

Drawback: The max operator in DQN, uses the same values both to select and to evaluate an action. This makes it more likely to select overestimated values, resulting in over-optimistic value estimates.

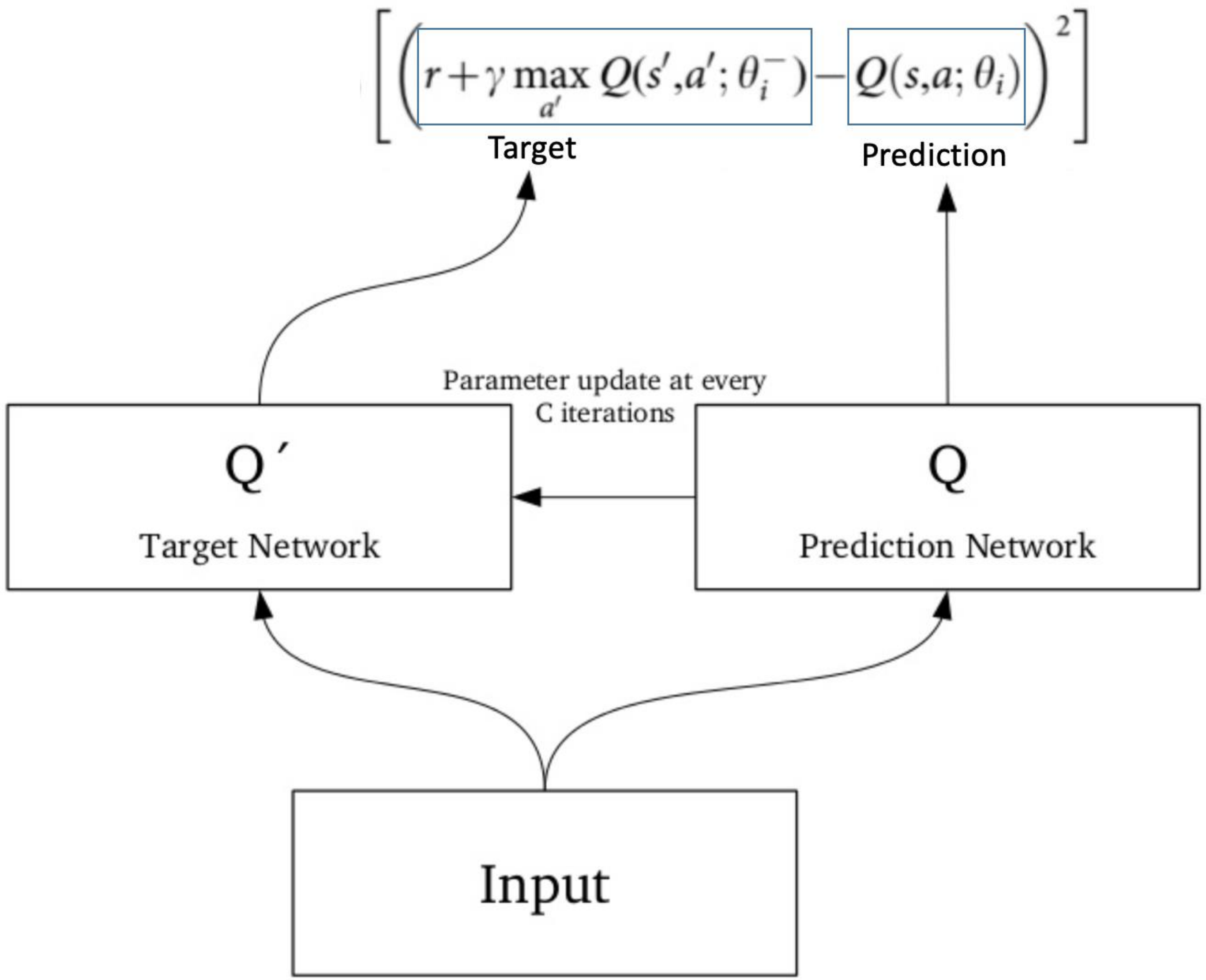
Double DQN

- Double DQN uses two networks to decouple the action selection from the target Q value generation.

    1. DQN network to select what is the best action to take for the next state (the action with the highest Q value).

    2. Target network to calculate the target Q value of taking that action at the next state.

- At every Tau step, we copy the parameters from our DQN network to update the target network.



Our Work

- From the prior experiments and study on various Deep RL algorithms, We have applied Dueling Double Deep Q-network (D3QN) with Prioritized experience replay.
- Explanations as follows
  - Dueling DQN
  - Experience Replay
  - Prioritized experience replay
  - D3QN training process

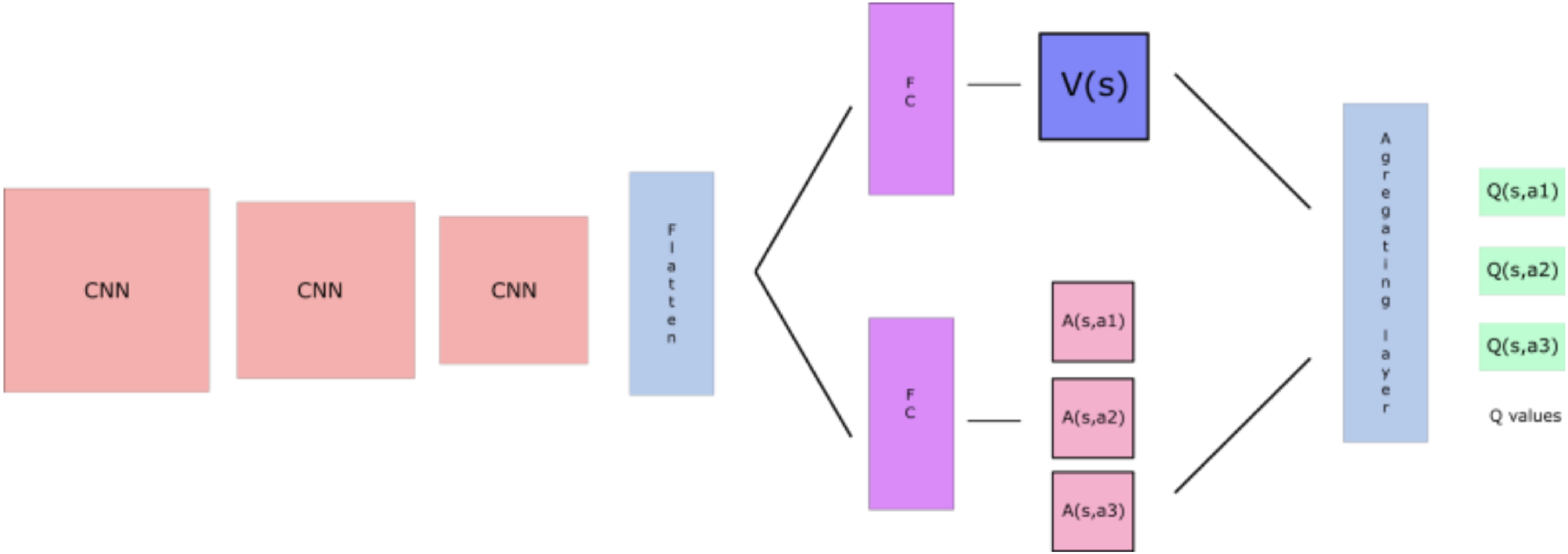
Dueling DQN

- Dueling DQN has two estimator, one estimates the score of current state, another estimates the action score.
- Dueling architecture to calculate Q values. Q-values correspond to how good it is to be at that state and taking an action at that state Q(s,a)
- Q(s,a) is the sum of:

- V(s) - the value of being at that state

- A(s) - the advantage of taking that action at that state (how much better is to take this action versus all other possible actions at that state).

$Q(s,a) = V(s) + A(s,a)$ **->** Advantage function



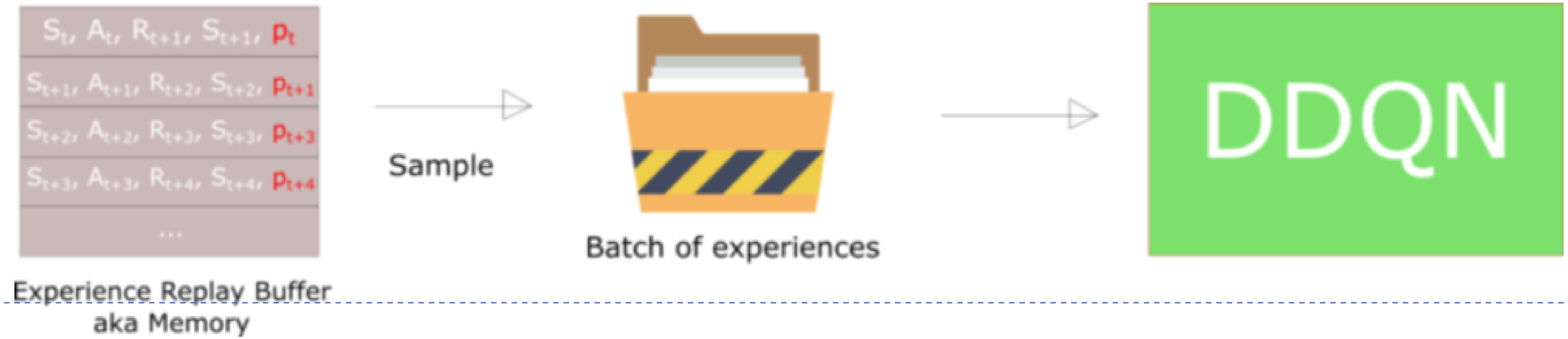
Dueling Deep Q-Network Architecture

Experience Replay

- Experience replay is used to handle two things:
  - Avoid forgetting previous experiences
  - Reduce correlations between experiences
- At each time step, we receive a tuple - state, action, reward, new_state (s,a,r,s')
- Replay Buffer stores experience tuples while interacting with the environment, and then we randomly sample a small batch of tuple to feed our neural network

Prioritized Experience Replay

- Some experiences may be more important than others for our training, but might occur less frequently (entropy)
- We compute the priority using the frequency of occurance
- Priority is stored with each experience in replay buffer.
- To avoid bias and overfitting, we use random minibatch of replay buffer to compute the loss



Training steps in Double Dueling Deep Q Network (D3QN)

1. Source and Target networks are Dueling DQN networks – 3 CONV2D networks with Relu activation
2. The training method is used to train our q network for q values.
3. Update target network every time after q network is trained for tau times.
4. The Q values are predicted for currents state and next states by q network and target network respectively.
5. From the frequency of occurance of state value, the prioritized experience replay computes the priority and stores with (s,a,r,s')
6. Then we sample batch of experiences. These sample batches are tuple array of states, actions, rewards, next states, done and priority variable.(s,a,r,s',p)
7. Compute Temporal Difference Error for loss computation
8. Then we calculate the batch index and perform the update operation. **Note that this update uses a double DQN update i.e instead of a max of q value of next state, we put q value predicted by the target network for action that has maximum q value according to q network in the next state.**
9. The update equation is multiplied by done variables because for terminal state q value is always zero.
10. Then we train the model, update epsilon, and increment train step.

$$\Delta w = \alpha[(R + \gamma \, max_a \, \hat{Q}(s', a, \boxed{w^-})) - \hat{Q}(s, a, \boxed{w})] \, \nabla_w \hat{Q}(s, a, w)$$

Change in weights

learning rate

Maximum possible Qvalue for the next_state (= Q_target)

Current predicted Q-val

TD Error

Gradient of our current predicted Q-value

At every T steps:

$$w^- \leftarrow w$$

Update fixed parameters

Temporal Difference Error computation (Loss function) and backpropagation

Explaining hyperparameters

- **SKIP FRAME**: Agent takes action on every k- frame instead of every frame, the last action is repeated on skipped frames.
- **ACTIONS**: Flappy bird has two actions: 1) Do Nothing 2) Flap.
- **GAMMA**: Gamma is the discount factor. It quantifies how much importance we give for future rewards. Gamma varies from 0.95 to 0.99.
- **ALPHA**: Alpha is the learning rate.
- **EPSILON**: Used for Exploration and Exploitation. The agent takes random actions for probability ε and greedy action for probability (1-ε).
- **SYNC_TARGET_FRAME** (Tau): We update the target network with the DQNetwork in every tau step.
- **EXPERIENCE_BUFFER_SIZE**: Size of the experience buffer.
- **MEAN_GOAL_REWARD**: The max reward an agent gets for all good action(positive reward) is +1. The mean goal reward is a programmer's choice to train the agent any value greater than or equal to +1.
- **BATCH_SIZE** : Neural network training batch size.

Experiment: Hyperparameter Tuning

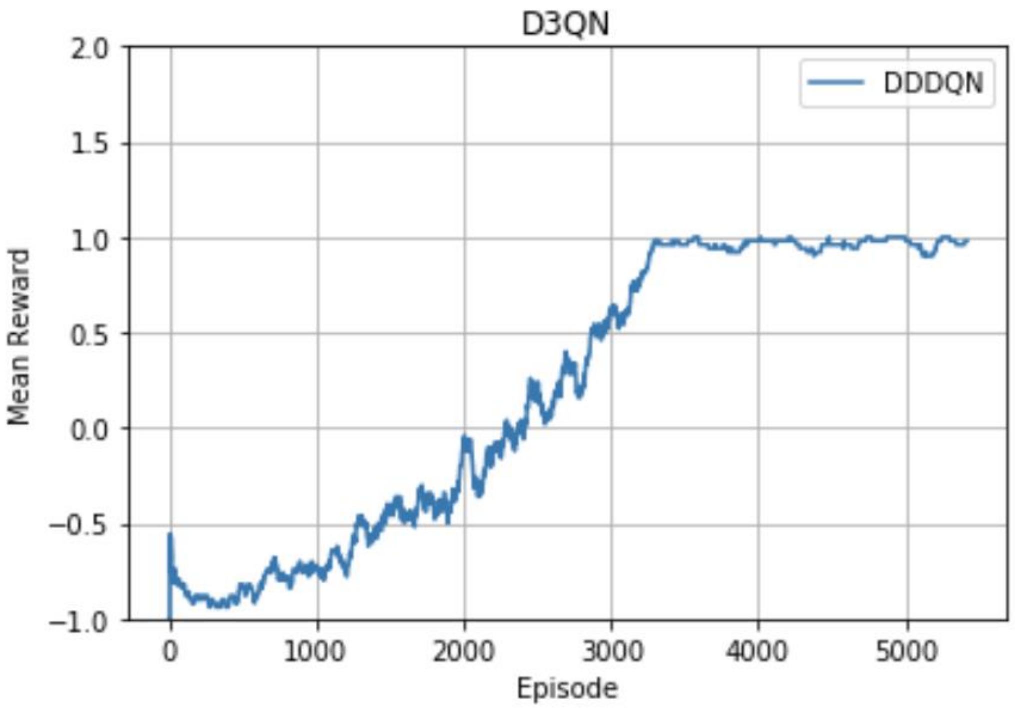
No	Hyperparameter	Value	Result	Convergence	Attempts Required in 3mins
1	SYNC_TARGET_FRAMES	15	GAME : 6680 | EPSILON : 0.0058 | MEAN REWARD : 1.0	YES	2
2	SYNC_TARGET_FRAMES	60	GAME : 23000 | EPSILON : 0.0010 | MEAN REWARD : 0.98	NO	
3	SYNC_TARGET_FRAMES	90	GAME : 6433 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	10
4	GAMMA	0.95	GAME : 5070 | EPSILON : 0.0010 | MEAN REWARD : 1.0	NO	
5	GAMMA	0.98	GAME : 5000 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	2
6	EXPERIENCE_BUFFER_SIZE	500	GAME : 16790 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	8
7	EXPERIENCE_BUFFER_SIZE	8000	GAME : 8210 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	3

Experiment: Hyperparameter Tuning

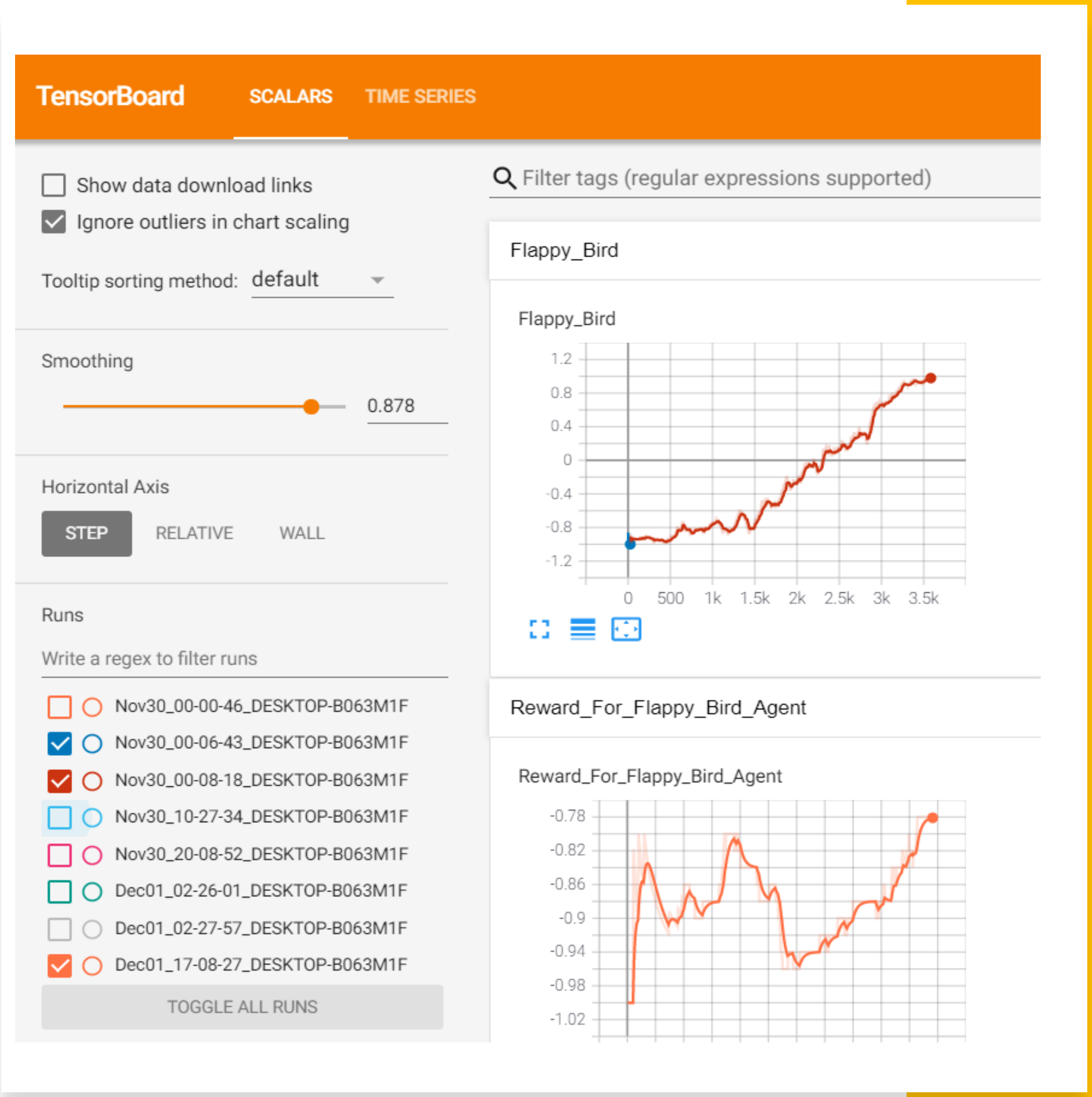
No	Hyperparameter	Value	Result	Convergence	Attempts Required in 3mins
8	LEARNING_RATE	1e-6	GAME : 25000 | EPSILON : 0.0010 | MEAN REWARD : .69	NO	
9	LEARNING_RATE	1e-2	GAME : 8735 | EPSILON : 0.0010 | MEAN REWARD : -1.0	NO	
10	SKIP_FRAME	5	GAME : 15015 | EPSILON : 0.0010 | MEAN REWARD : 0.4	NO	
11	SKIP_FRAME	1	GAME:3375|EPSILON: 0.0010| MEAN REWARD : 0.98	YES	4
12	BATCH_SIZE	16	GAME : 4030 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	2
13	BATCH_SIZE	64	GAME : 4030 | EPSILON : 0.0010 | MEAN REWARD : 1.0	YES	2

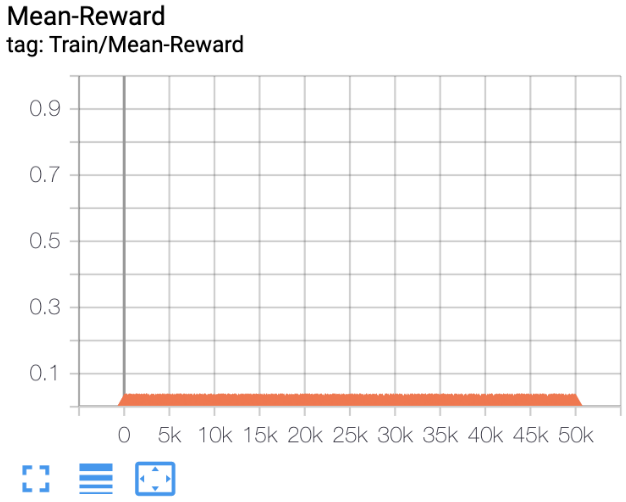
Best Performing Model

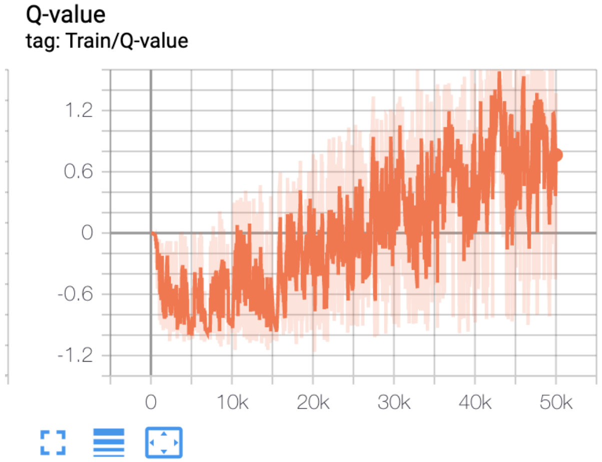
Hyperparameter	Value	Result
EXPERIENCE_BUFFER_SIZE	2000	GAME : 5225 | EPSILON : 0.0010 | MEAN REWARD : 1.0
STATE_DIM	4	
ACTIONS	[0,1]	
GAMMA	0.99	
EPSILON_DECAY_FRAMES	(10**4)/3	
MEAN_GOAL_REWARD	1	
BATCH_SIZE	32	
SYNC_TARGET_FRAMES	30	
LEARNING_RATE	1e-4	
SKIP_FRAME	2	

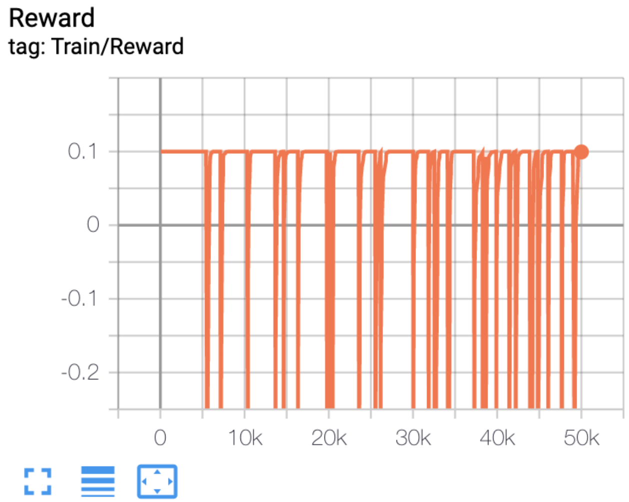


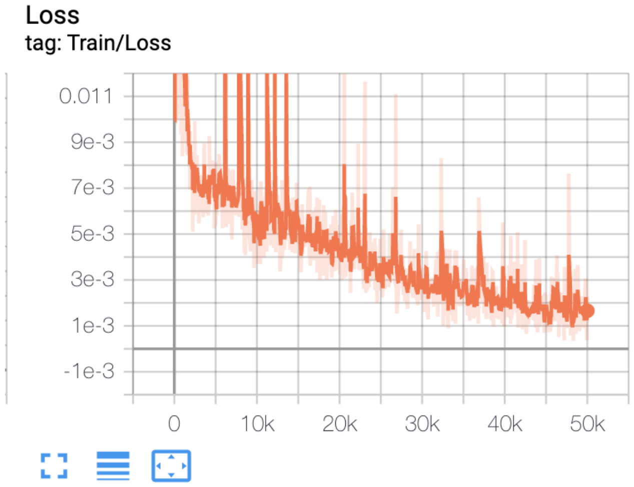
TensorBoard





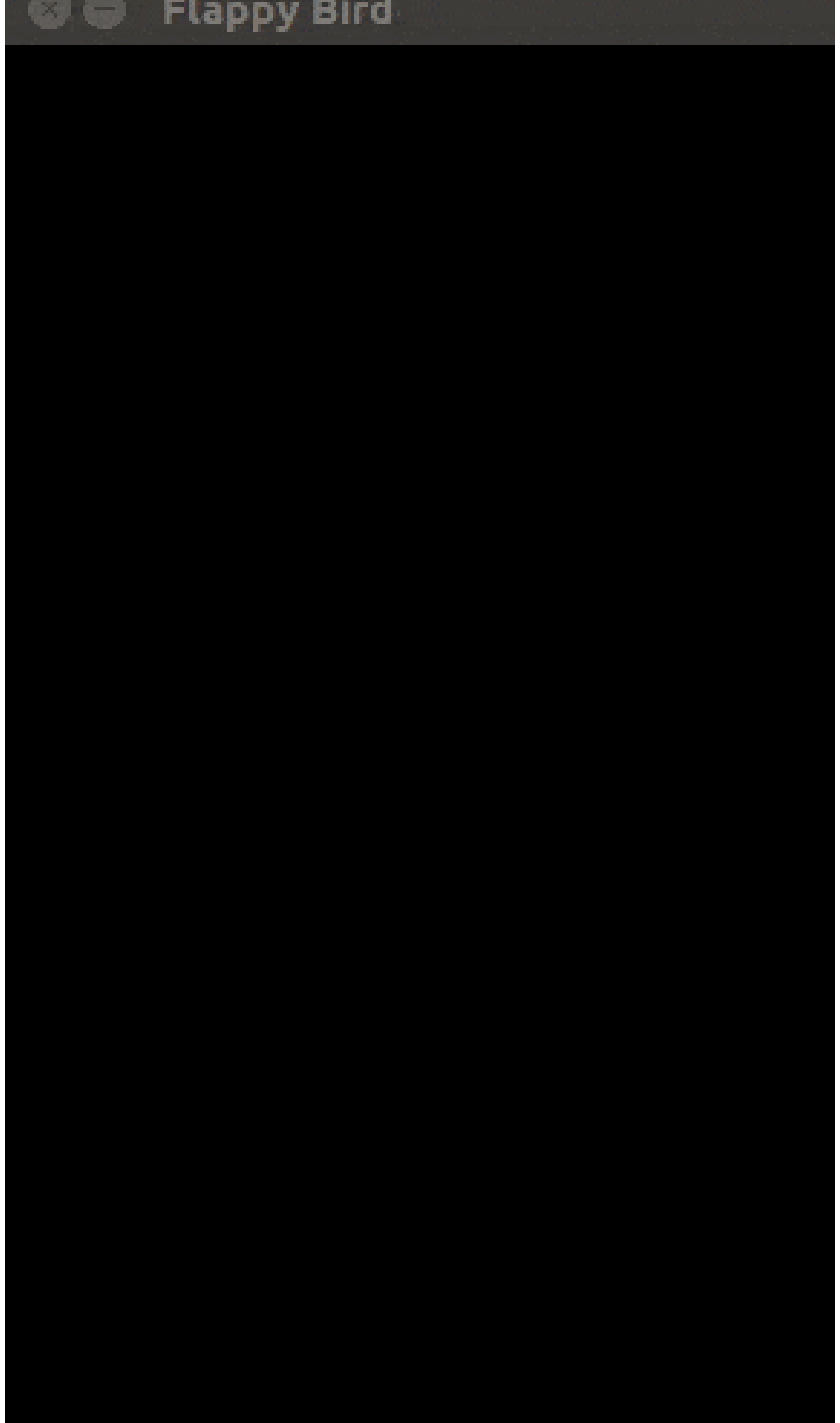






Comparison study with DQN

We tried experimenting flappy bird agent with Deep Q Network. After training for 50K iterations, the network didn't converge.



Demo

Key learnings and challenges

Key Learnings

- We explored some Deep Q Network – DQN, DDPG
- We understood Flappy bird agent's action and state space
- Training an end-to-end deep reinforcement learning algorithm
- Understanding and Fine-tuning hyperparameter to get the best model

Challenges

- Flappy bird agent is not present in open gym environment, hence we used pygame, gym_ple or custom python code from opensource to setup the agent for programming.
- Training time with GPU machine is also high, hence experiments took much time to complete
- Very few Deep RL algorithms were applied to train this agent

Project Management

- Github: https://github.com/s-c-soma/RL_Project_FlappyBird_D3QN
- Trello: https://trello.com/b/8CrK9MHe/rl-the-mean-squares

Team members Contribution

- **Pranav Lodha**
  - Literature study on various DQN
  - Understanding prioritized experience replay
  - Inference pipeline with best model
- **Subarna Chowdhury Soma**
  - Understanding D3QN
  - Exploring various research works in D3QN
  - Experiments with hyperparameters
- **Jeyasri Subramanian**
  - Exploring various DQN architecture and design architecture
  - D3QN environment setup
  - Comparison study with DQN
- **Team**
  - Report writing
  - Presentation
  - Demo
- Each team member shared the knowledge with other team members about literature study and architecture

References

- https://www.fromkk.com/posts/using-ddqn-to-play-flappy-bird/
- https://towardsdatascience.com/use-reinforcement-learning-to-train-a-flappy-bird-never-to-die-35b9625aaecc
- https://towardsdatascience.com/practical-reinforcement-learning-02-getting-started-with-q-learning-582f63e4acd9
- http://cs229.stanford.edu/proj2015/362_report.pdf
- https://www.freecodecamp.org/news/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8/
- https://www.freecodecamp.org/news/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682/
- https://markelsanz14.medium.com/introduction-to-reinforcement-learning-part-4-double-dqn-and-dueling-dqn-b349c9a61ea1
- https://arxiv.org/pdf/1511.06581.pdf
- https://arxiv.org/abs/1509.06461

Thank you

Questions