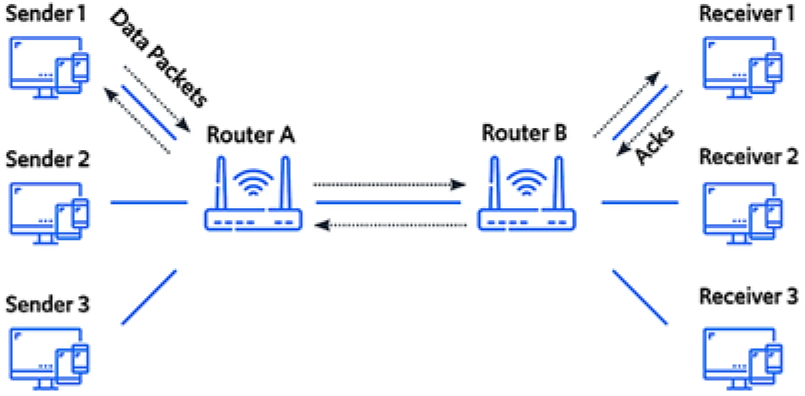
Optimal Path Routing Using Reinforcement Learning

Team Members:

Navodita Mathur
Sharika Loganathan
Vivek Kumar

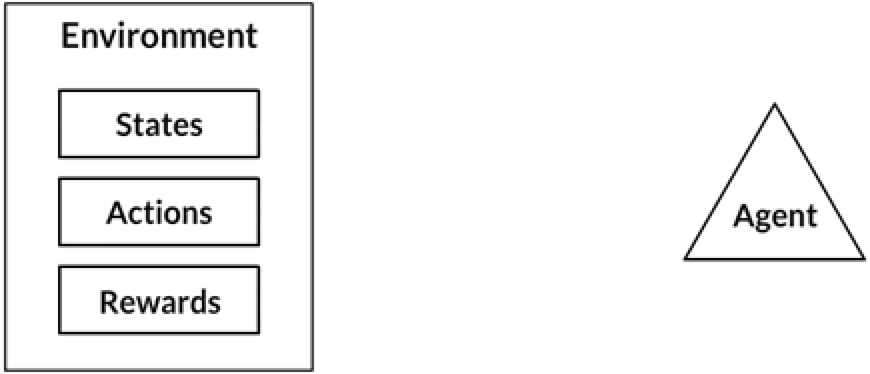
Problem

- Routers use routing algorithms like Dijkstra, Bellman-Ford, and Ford-Fulkerson to find the best route from a source node to a destination node through multiple switches and routers.
- These algorithms have drawbacks, such as being complex, taking a long time to process, and not being dynamic enough to consider parameters like traffic congestion and queue size.
- In internet congestion control, the goal is to modulate traffic sources' data-transmission rates to efficiently use network capacity. Reinforcement learning (RL) can be used to train deep network policies that capture intricate patterns in data traffic and network conditions, improving performance.

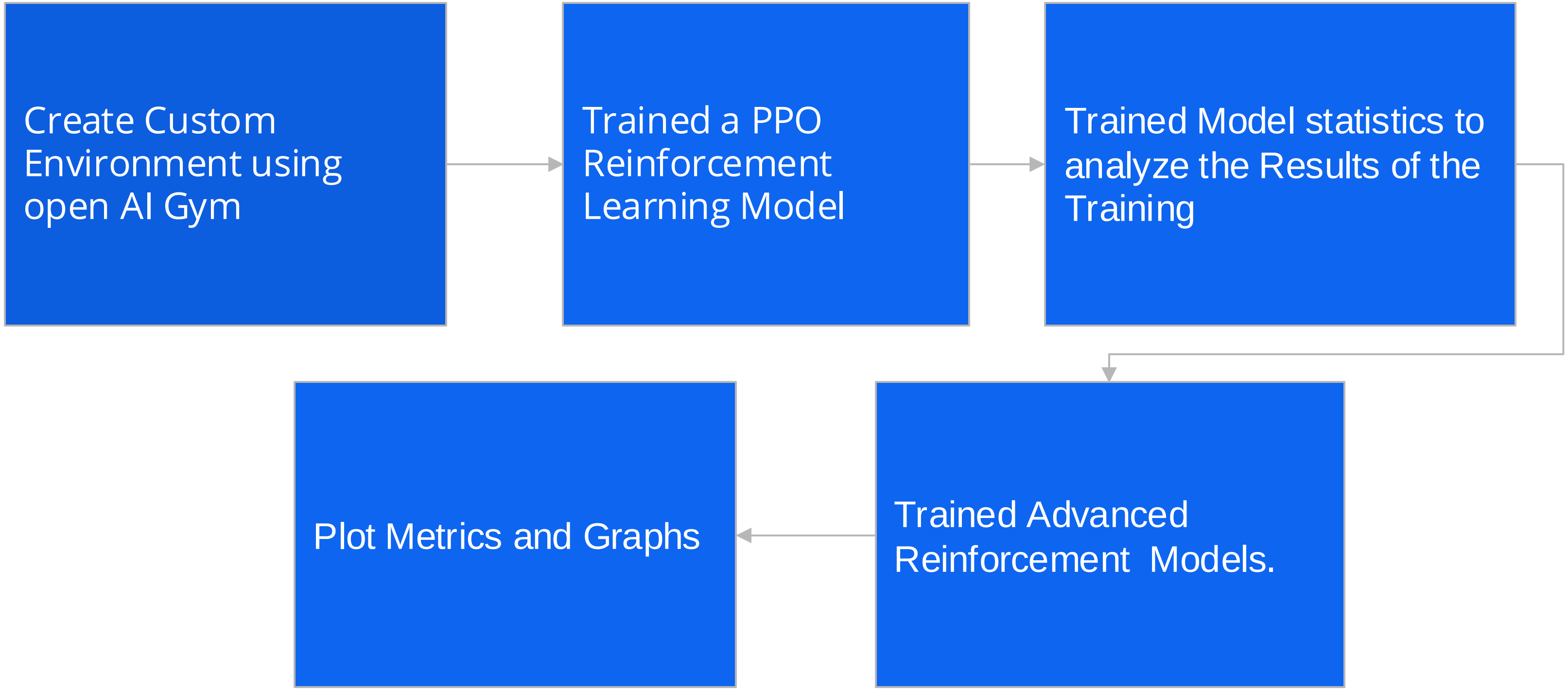


Why Reinforcement Learning Algorithm ?

1. Adaptive behavior
2. Scalability
3. Robustness
4. Efficiency



Bird's Eye View Approach (Implementation Approach)



Create Custom Environment using open AI Gym

Trained a PPO Reinforcement Learning Model

Trained Model statistics to analyze the Results of the Training

Trained Advanced Reinforcement Models.

Plot Metrics and Graphs

Environment Creation

OpenAI gym : An environment for developing and testing learning agents. It is focused on reinforcement learning agent.

We have created an Environment using openAI gym.

States : States is comprised of a state vector which has 3 elements describing the following:

1. Latency Gradient: the derivative of latency with respect to time
2. Latency Ratio :the ratio of the current MI’s mean latency to minimum observed mean latency
3. Sending Ratio:the ratio of packets sent to packets acknowledged by the receiver

State at Time **T ,** $s_t = (v_{t-(k+d)}, \ldots, v_{t-d})$

Here, St is the state vector with components being a list of a list.Each element of the state vector is a list having the 3 metrics described above.

k is the window time from which each list has to be used in St

Actions:

Actions are changes to sending rate(at)

where , $x_t = \begin{cases} x_{t-1} * (1 + \alpha a_t) & a_t \geq 0 \\ x_{t-1} / (1 - \alpha a_t) & a_t < 0 \end{cases}$

Rewards:

The Reward Function is $10 * throughput - 1000 * latency - 2000 * loss$

Environment Creation

Environment Specification:

The agent is the transmitter of traffic in our concept, and its activities lead to changes in sending rates. We use the concept of monitor intervals (MIs) to define this.Time is split into segments that are successive. The sender can alter its transmission rate xt at the start of each MI t, which remains constant during the MI.

Algorithms Used

Proximal Policy Optimization (PPO):

- Reinforcement learning algorithm that optimizes a neural network policy to maximize expected rewards
- Uses surrogate objective function to prevent large policy updates that could cause instability or performance degradation
- Example application: training an agent to play a game like Atari's Breakout

Hindsight Experience Replay (HER):

- Technique used to improve efficiency of learning from failed experiences in reinforcement learning
- Relabels unsuccessful experiences as successful by changing agent's goal to achieved state instead of original desired state
- Example application: training an agent to solve a robotic manipulation task like stacking blocks

Training Metric Analysis

This graph explains the Training Reward versus Training Episode for different values of **λ.**

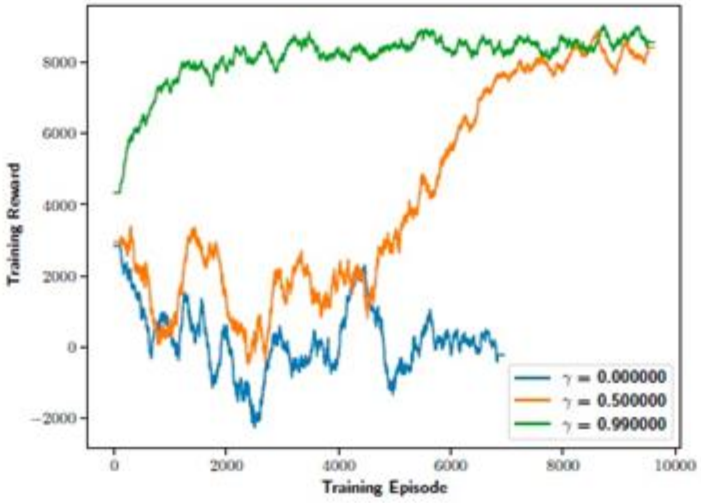
λ is the discount factor for the rewards

Discount factors are a way to balance immediate rewards against future rewards in a reinforcement learning problem. They represent the relative importance of immediate rewards versus rewards that may come later in the agent's interaction with the environment.

The choice of discount factor can have a significant impact on the behavior of an agent in a reinforcement learning problem. A high discount factor will prioritize long-term rewards, while a low discount factor will prioritize immediate rewards.

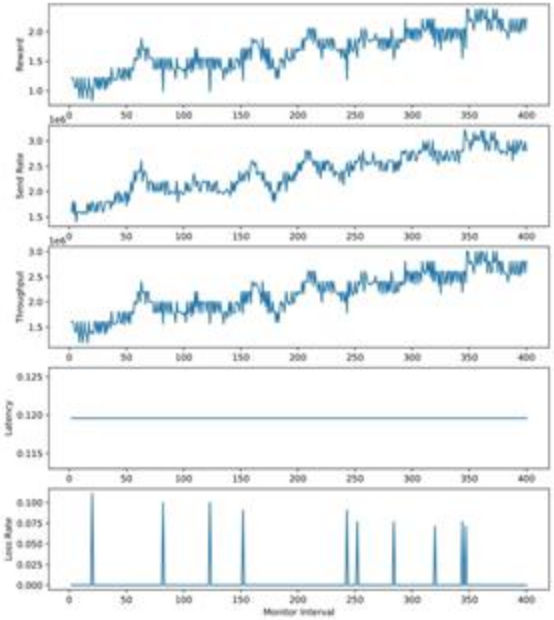
As we can see in the Figure **λ= 0.99** increases the training reward quickly but we have chosen **λ= 0.5.**

Our model requires some warmup time before the loss metrics starts to kick in, This is why **λ= 0.5**. Gets to the gradient of the maximum training reward but does so progressively and hence does not overestimate the efficiency.

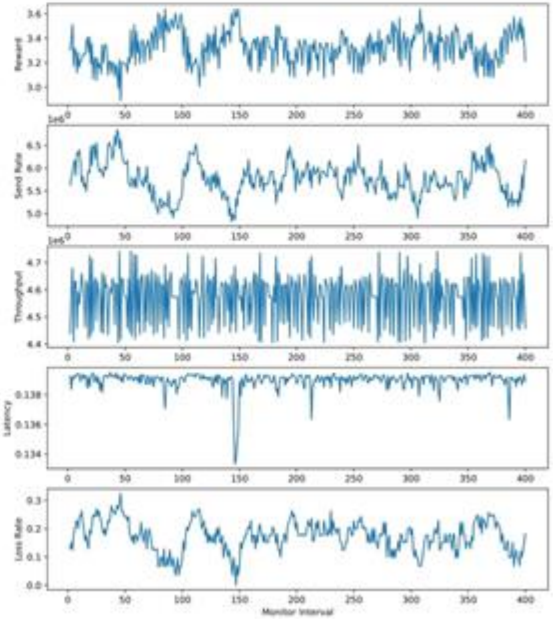


Evaluation Results

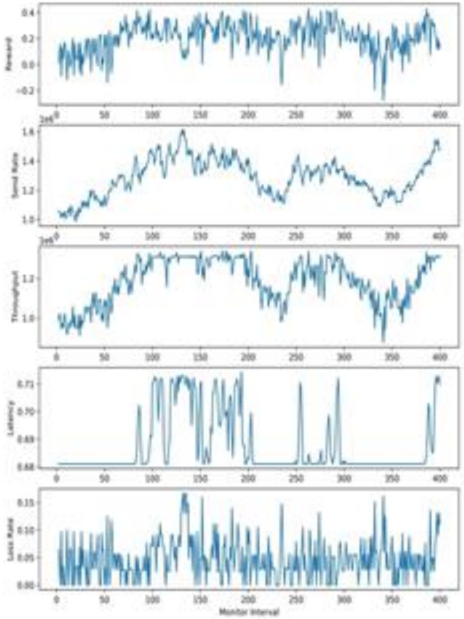
Evaluation Metrics at iteration 100



Evaluation Metrics at iteration 5000



Evaluation Metrics at iteration 9900



Evaluation Results Discussion

As shown in the previous slide,

The Graphs tell us about the different evaluation metrics at Iteration 100,5000 and 9900.

The progress of the latency metric can help us understand the evaluation.

- Iteration 100 - We can observe that the latency is quite low and consistent.Because the model is still in its early stages, the routers haven't had a chance to populate the queues. This permits the packets to be transmitted without interruption.
- Iteration 5000 - In this case , the routers had time to add packets to the queues but the model is not completely trained. Hence, we observe a high latency as the policy is not optimal.
- Iteration 9900 - Now, although the routers are filling up their queues frequently but the model has had ample time to train. This provides an optimal policy which results in low latency.

Hence the Iteration 9900 is the most representative of real world scenario and the policy we obtained at that Iteration is the optimal one as it started to keep the latency low.

Similar conclusions can be drawn for other metrics as well.

Task Allocation

Task allocation

Vivek:
1. Gym Environment Code
2. Environment Model creation
3. Demo presentation

Sharika:
1. Model training Code
2. Training Performance Metric analysis and evaluation
3. Project presentation

Navodita:
1. GUI design
2. Model performance analysis and code
3. Project presentation.

Resources

https://computer.howstuffworks.com/routing-algorithm.htm

https://github.com/openai/gym

https://pypi.org/project/PyQt5/#:~:text=PyQt5%20is%20a%20comprehensive%20set,platforms%20including%20iOS%20and%20Android.

https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/

https://www.youtube.com/watch?v=eClQWW_gbFk

https://pythonprogramming.net/q-learning-reinforcement-learning-python-tutorial/