Vidyadhar

by Vidyadhar Vidyadhar

Submission date: 12-Jun-2023 02:13PM (UTC+0530)
Submission ID: 2114358572
File name: Report_Payment_Detailed.pdf (7.87M)
Word count: 17186
Character count: 90463

STRATEGIZING TO SELECT THE OPTIMAL PAYMENT TYPE USING REINFORCEMENT LEARNING ALGORITHMS

Vidyadhar Bendre Saranya A (Project Guide)

ABSTRACT

Modern commerce has increasingly dependent on digital payments since they make it easier for customers and companies to transact. To prevent hiccups and discomfort, it is essential to choose the right payment option. In order to improve transaction success rates, reduce fees and processing times, and avoid payment process interruptions, this work suggests using reinforcement learning (RL) to optimize payment method selection.This survey paper's goal is to present a thorough overview of online payment systems in the context of RL and their use in directing various payment methods. It starts out by introducing different RL algorithms that have been suggested for payment routing, including Q-learning, DQN, DDQN, and replay DDQN. The exploration-exploitation trade-off in RL-based payment routing, high-dimensional state space, and non-linear dynamics are some of the issues that are addressed in this work. The research also investigates various RL-based approaches for optimal payment gateway routing, including the application of neural networks. It summarizes the most recent algorithms for RL-based payment gateway routing in the conclusion and makes recommendations for further research. For researchers, engineers, and practitioners interested in RL-based routing as the best payment gateway, this survey article aspires to be a useful resource.

***K*eywords** Reinforcement Learning · Routing payment · Optimal payment type

1 Research

1.1 RL Foundations

1. Supervised, Unsupervised and Semi-supervised Learning *Learning from labelled and unlabeled data is a component of the machine learning methods of supervised and unsupervised learning, respectively* In order to train a machine learning model on a labelled dataset, supervised learning requires that each input example be labelled with the desired result. This can be useful if the intended behaviour is well-defined and there are sufficient training data to build a robust model. However, choosing the appropriate payment gateway can be difficult, especially in dynamic situations with a high level of unpredictability. It may be difficult to specify the desired behaviour or to create a sufficiently large and diverse labelled dataset for supervised learning. Overall, even though supervised learning and unsupervised learning can be useful for a variety of tasks, they may not be well-suited for routing the best payment gateway due to the complexity and continuous nature of the target variable as well as the need to make specific predictions or decisions in real-time.
2. Semi-supervised Learning: *Building models with machine learning approaches like semi-supervised learning need both labelled and unlabeled data.* This approach involves training a model on a small sample of labelled data before using it to label a larger sample of unlabeled data. The model may learn from more data as a result, improving its accuracy. Algorithms for semi-supervised learning include, in some cases:
   - Co-Training: In this method, the unlabeled data is labelled using two models that have been trained on various interpretations of the data. The models are then trained using both the previously tagged data and the newly labelled data.
   - Self-Training: This algorithm labels unlabeled data using a model that has been trained on labelled data. The model is then retrained using the combined labelled and newly labelled data.
   - Graph-Based Approaches: This approach visualises the data as a graph, and propagates the labels across the graph using graph-based algorithms.
   - Generic models: The distribution of the data is taken into account by these models, which then use that information to create new, synthetic data points that may be used to train the model.

   Semi-supervised learning may be useful when there aren't enough labelled data or when labelling a lot of data would be expensive or time-consuming. It can also be useful for boosting the accuracy of models when the amount of labelled data is negligible in relation to the amount of unlabeled data. It's possible to utilise semi-supervised learning to help select the optimal payment gateway, although it's more likely that other machine-learning methods would be more widely used for this purpose. A common technique for selecting the optimal payment gateway is supervised learning, in which a model is trained on a sizable dataset of labelled examples that show the correct actions to take in various situations. Semi-supervised learning can be combined with supervised learning to improve model accuracy by using a smaller amount of labelled data to train the model and a greater amount of unlabeled data to label. To choose the best gateway in a payment environment, other *approaches, such as reinforcement learning or imitation learning*, would also be utilised to aid in the selection of the ideal gateway in a payment context.
3. Imitation learning: *The training of a model to replicate the activities of an agent who can select the best payment gateway is a component of the imitation learning method to machine learning.* The following steps explain how to utilise imitation learning to select the ideal payment type:
   (a) creation: Create a dataset of payment transaction data by either engaging e-commerce service providers or creating the data through distributions.
   (b) Preprocessing: Data preprocessing and labelling may be required in order to get the gathered data ready for machine learning model training.
   (c) Training: Create a machine learning model and train it using the labelled data to forecast the most effective gateway for the input data. Using this model, the best payment environment gateway can then be selected.
   (d) Test and improve the model: To evaluate the trained model's performance and identify any defects, it can be tested on new data. The model can then be enhanced and retrained based on the results of the testing.
4. Reinforcement Learning: *Reinforcement learning, or RL for short, is a machine learning method that teaches an agent to act in a way that maximises rewards in its environment. With RL, we can train the agent to figure out how to select the optimal payment gateway in every given situation.* Following these guidelines will help you select the ideal payment environment gateway using RL:
   (a) The Open AI Gym environment has been enhanced by defining the reward, observation space and action space.

(b) The success rate, processing time, and transaction fees are used to define reward.

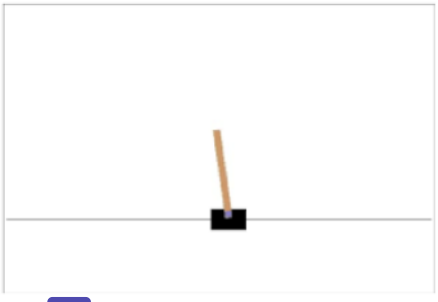
(c) To teach a model the control policy: We have defined five agents. Qlearning Agent, DQN, DDQN Agent, Duelling DDQN AgentA, MORL Duelling DDQN agent and MORL Thompson Duelling DDQN Agent. These agents can be used to train a control strategy that maximizes the reward signal. By interacting with the payment environment and being paid for every activity, we may accomplish this. The RL algorithm will eventually be able to decide how to maximize the expected return.

(d) Test and improve the model: To evaluate the trained model's performance and identify any defects, it can be tested on new data. The model can then be enhanced and retrained as appropriate based on the results of the testing.

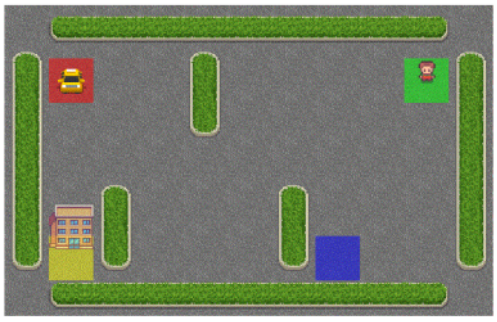
Overall, by instructing an agent on the optimal course of action that will result in the most rewards, RL can be a useful strategy for selecting the payment environment's gateway. It is essential to thoroughly examine the architecture of the environment, the reward, and the RL algorithm in order to achieve better results.

5. Agent:An agent A computer learning model called Agent has been trained to carry out tasks in a way that will maximize a reward. To use an example, the agent is frequently an algorithm (Q-Learning, DQN, DDQN, or Duelling DQN) that has been trained to select the ideal payment gateway type in a particular situation. The agent interacts discretely (t) in timesteps with an environment (E).

(a) A Steering wheel (Algorithm that helps to take right action - to keep at zero degree or rotate by theta angle in order to follow the right path)



(b) An agent that takes actions in the environment in order to achieve a certain goal or objective



(c) Agent that controls the actions of the taxi cab in order to achieve a certain goal or objective

(d) Agent that controls the actions of a character moving on a grid of squares

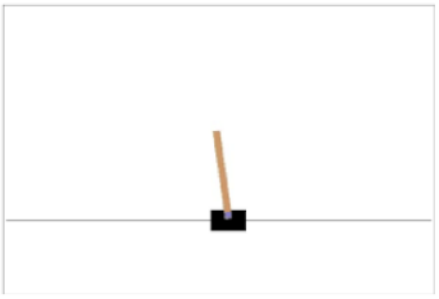
Figure 1: The figure illustrates the four different agents

6. Environment: *The environment* includes the state that includes the amount, success rate, processing time, transaction charges and status. The Environment receives Action (At) and emits observation (Ot). The Environment can be deterministic or stochastic. In the case of deterministic, there is only one single observation for a given history and action. In stochastic, there are many observations and rewards for a given history and action.

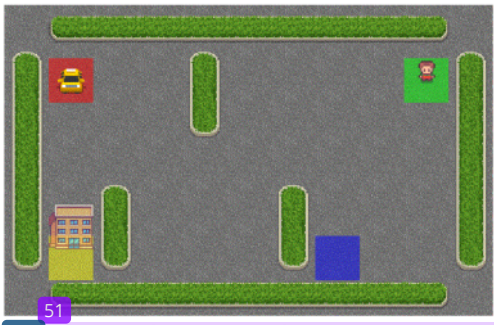
7. History:
   The History: A chronicle of earlier experiences or events is referred to as 'The History'. History in the context of RL can refer to the progression of deeds, conditions, and benefits that have taken place over time in a certain setting.
   The order in which past tasks were completed, such as choosing a payment environment gateway, as well as the system's states at various times and the incentives the agent earned for various actions.
   In order to analyze the prospective results and benefits of various actions, the agent may use the history of its previous experiences to spot patterns and trends that can help it make more educated decisions. A series of historical observations (Ot), deeds (At), and rewards (Rt) make up history (Ht). As (o1, a1, r1), (ot-1, at-1, rt-1),...(to, at, rt), this can be represented.

8. Action:
   The action is a decision that an agent makes in reaction to its environment. The action taken determines how quickly things move from one state to the next. (such as choosing one of the seven payment types).

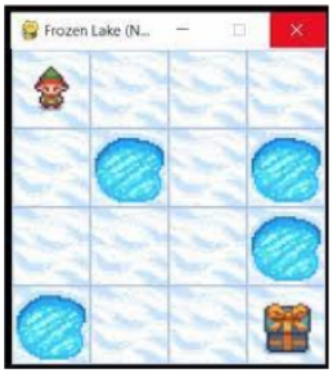
(a) Rotation of the steering wheel and acceleration (steady, increase or decrease)



(b) (0: apply force to the left 1: apply force to the right)



(c) (0: move south, 1: move north, 2: move east, 3: move west, 4: pickup passenger, 5: drop off passenger, 6: do nothing/idle)



(d) (0: move left, 1: move down, 2: move right, 3: move up)

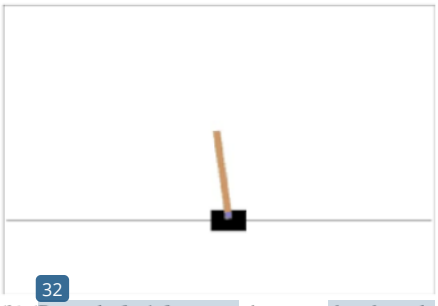
Figure 2: The figure illustrates the four different actions

9. State: *The state* contains details about the current state or situation and its surroundings, such as the success rate, length of time taken to complete, sum, transaction fees, and status. State(St). Both completely and partially visible environments can be found in the environment. Agents may find it more difficult to operate in partially visible environments since they must have an internal model of the environment's state and use it to guide their decisions. This can be challenging since the agent may need to process and update its internal representation regularly in a complex and dynamic environment.
In fully observable environments, the agent need only use the environment's current state to decide rather than maintaining an internal representation.
The information utilized to decide what happens next is the state. It is classified as a result of history.

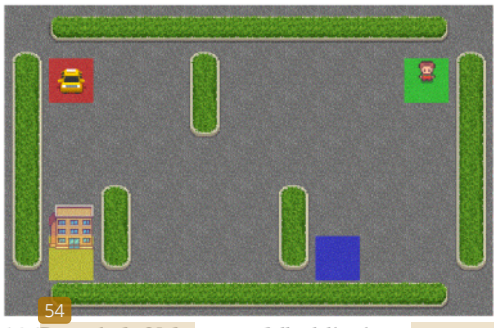
$$St = f(Ht)$$

10. Rewards: *The rewards* is an incentive system that uses rewards and penalties to communicate to the agent what is right and wrong. In real life, an agent's objective is to maximize overall rewards. The agent gains the ability to occasionally forgo immediate rewards in favour of maximizing overall rewards.
The correct or positive reward such as choosing the best gateway that maximizes the success rate and minimizes both processing time and transaction fees. On the other side, the incorrect or negative reward is connected to unwanted or harmful results or behaviours, such as failing gateway status.

(a) (Rewards should be zero until it reaches to the destination correctly, Only upon reaching the destination should receive the positive reward)



(b) (Reward of +1 for every time step that the pole remains upright, and a reward of 0 for every time step that the pole falls or the episode terminates)



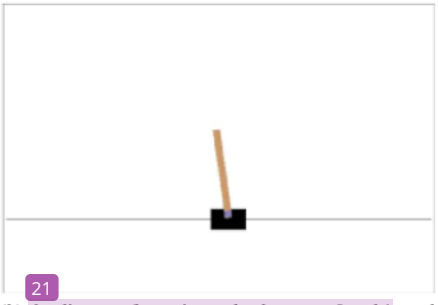
(c) (Reward of +20 for successfully delivering a passenger to the destination, a reward of -10 for illegal actions (e.g. moving on a wall or dropping off a passenger not in the destination), a reward of -1 for each time step, and a reward of -10 for each time step spent with an undelivered passenger in the cab)

(d) (reward of +1 for reaching the goal and a reward of 0 for all other states and actions)

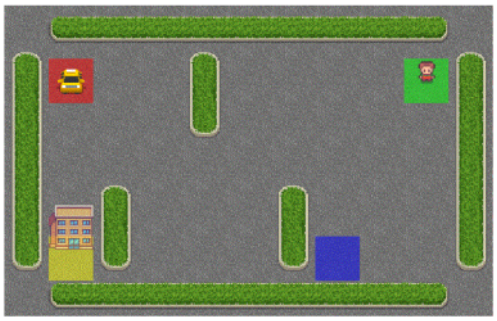
Figure 3: The figure illustrates the four different rewards

11. Discount Factor:
Discount Factor is a factor that helps to get the current state value by multiplying it by future rewards. It is denoted by γ a value ranging between 0 and 1, with higher values indicating a greater emphasis on long-term rewards. The discount factor would be used to balance the importance of immediate rewards (for example, choosing a gateway with low processing time and transaction charges) versus future rewards in the context of a payment environment where the goal is to choose the best gateway to minimize processing time and transaction charges and maximize the success rate. (e.g., choosing a gateway with a high success rate).

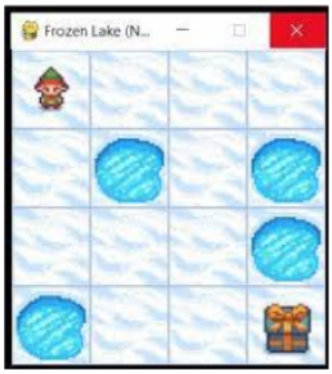
(a) (if gamma is close to 1, the algorithm will prioritize long-term rewards and try to keep the car on the road as much as possible, while if gamma is close to 0, the algorithm will prioritize short-term rewards and try to reach the destination as quickly as possible)



(b) the discount factor is a value between 0 and 1 used in the calculation of expected cumulative reward for an agent's actions. It determines the importance of future rewards versus immediate rewards. A small discount factor will make the agent focus on short-term rewards, while a larger one will make it focus on long-term rewards.



(c) an agent is tasked with picking up and dropping off passengers in a grid-like city while avoiding illegal actions such as parking at the wrong location and the discount factor can be a good tool for the agent to balance the trade-off between reaching the final destination as soon as possible and collecting more rewards for completing tasks like picking up and dropping off multiple passengers.



(d) (The discount factor can be used to balance the trade-off between reaching the goal as soon as possible and avoiding the holes, if the agent is given a high discount factor it will be more inclined to take riskier moves to reach the goal quickly but it will also be more prone to falling into the holes and vice versa.)

Figure 4: The figure illustrates the four different discount factors

In this instance, the discount factor ($gamma$) can be defined as a parameter that denotes the weighting of future rewards in relation to present rewards. The agent would be encouraged to select a gateway that might not have the lowest immediate rewards (i.e., processing time and transaction fees), but is more likely to result in higher long-term rewards (i.e., success rate), if there was a high discount factor in place. Even though it would not be the best option over the long run, a low discount factor would encourage the agent to select a gateway with the quickest processing times and lowest transaction fees. As a result, the discount factor can be utilized to balance current and future incentives in the payment environment as well as to direct the agent's choice of the best gateway. The exact aims and restrictions of the payment environment, as well as the particular reinforcement learning algorithm being employed, would determine the value of the discount factor.

12. Information State:

    The Information State The information state would typically include the current state of the payment system and any pertinent contextual information that the agent can use to make decisions in the context of a payment environment where the objective is to choose the best gateway to minimize processing time and transaction charges and maximize the success rate.
    The information state would typically include the current state of the payment system and any pertinent contextual information that the agent can use to make decisions in the context of a payment environment where the objective is to choose the best gateway to minimize processing time and transaction charges and maximize the success rate.
    An Information state also known as Markov State contains all the useful information from history. A Markov state is typically defined as a system that undergoes transitions between a finite set of states over time, where the probability of transitioning from one state to another is only dependent on the current state and not on the previous states. This property is known as the Markov property. However, it is also possible to define a Markov state that takes into account the history or information state of the system. This can be accomplished by expanding the list of states to include not only the system's current state but also its former states and any pertinent details about its past.
    The definition of a Markov state allows us to take into account the history of the system's previous states when determining the probability of transitioning from one state to another.

13. Episode: *The episode* An episode is a series of steps made by an agent to choose the best payment gateway for a particular transaction in the context of a payment environment. Starting from a state known as the initial transaction amount, the user or merchant making the payment, and any other pertinent contextual information, the episode can proceed. The agent then performs a series of operations, such as requesting information from the various gateways regarding their success rates, transaction costs, and processing times, and choosing the best gateway based on both that gateway's current condition and any pertinent contextual data. At each stage, the agent is given feedback in the form of a reward, which may be determined by criteria such as the gateway's success rate, processing speed, and any associated transaction fees. When the transaction is finished, whether successfully or unsuccessfully, and the terminal state is attained, the episode comes to a conclusion. The agent now receives a final payment determined on the success or failure of the transaction. This reward may be positive if the transaction was a success or negatively if it failed or resulted in excessive transaction fees. In a payment environment episode is a series of decisions made by an agent to choose the best payment type for a certain transaction, beginning in an initial state and finishing in a terminal state. At each state, the agent is given feedback in the form of awards, and the final payout is determined by how the transaction turned out. Depending on the precise objectives and limitations of the payment environment, as well as the precise reinforcement learning algorithm being utilized, the actions and rewards may differ.

14. Return: *The Return* The return can be referred as the cumulative reward or overall reward obtained by an agent for a certain transaction. The success rate of the chosen gateway, the processing time of the chosen gateway, and any transaction fees incurred can all be used to determine the return for a particular transaction. The return should consider all pertinent elements that have an impact on the agent's performance in the payment environment and provide a numerical indicator of the agent's achievement of its objectives.

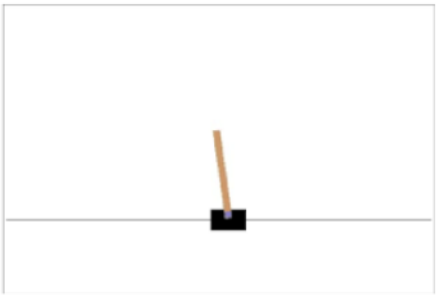
    An agent's expectation of a long-term payoff from a particular state or activity is measured by the return. It is referred to as the sum of the discounted future benefits that the agent anticipates receiving, with the relevance of future rewards in relation to immediate rewards being determined by the discount factor $gamma$. The value of a particular condition or action can be assessed using the return, and the agent's decision-making process can be directed by it.
    The following equation often serves as a representation of the return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$

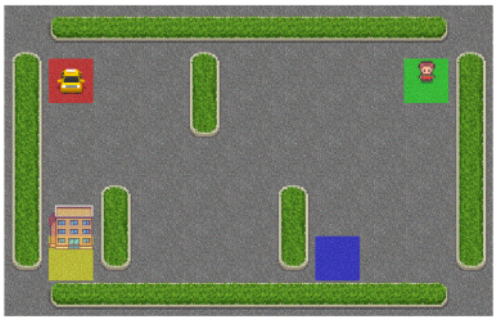
where,

- G_t is the return at time t,
- R_t is the reward received at time t and
- γ is the discount factor.

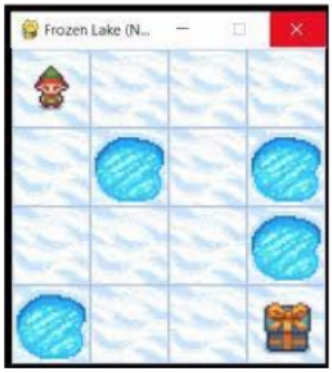
(a) (return for a given episode is calculated based on the distance travelled, the fuel efficiency of the vehicle, and any penalties incurred for collisions or other unsafe behaviours)



The agent controls a cart that moves left and right to balance a pole that is attached to it. The agent receives a reward of +1 for each time step that the pole remains balanced and a penalty of -1 for each time step that the pole falls down. The agent's objective is to balance the pole as long as possible and the discount factor helps to balance the trade-off between getting more rewards by balancing the pole for a longer time or taking more risks to balance the pole for a shorter time)



(c) (the return would be the sum of the rewards the agent would get for picking up and dropping off the passenger at the right location, avoiding illegal actions such as parking at the wrong location, and reaching the final destination as soon as possible)



(d) (the return would be the sum of the rewards the agent would get for reaching the goal and avoiding the holes. The agent's objective is to find the best path to reach the goal, which would have the highest return value and the discount factor helps to balance the trade-off between reaching the goal as soon as possible and avoiding the holes)

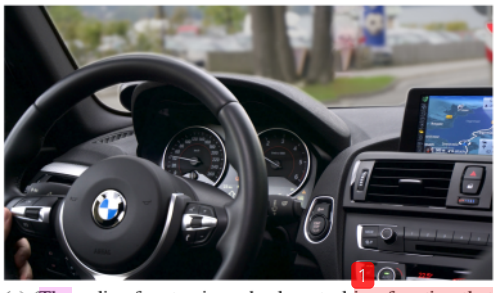
Figure 5: The figure illustrates the four different returns

15. Policy: *The policy* A function called the policy outlines the course of action to be taken in each state. It describes a mapping from states to actions and exhibits the agent's method of decision-making.It is denoted by

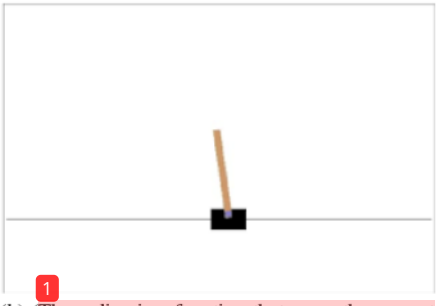
π.
Finding a policy that maximizes the expected benefit over time is the aim of an MDP. Several algorithms, including value iteration and policy iteration, can be used to accomplish this.

There are two main types of policies:

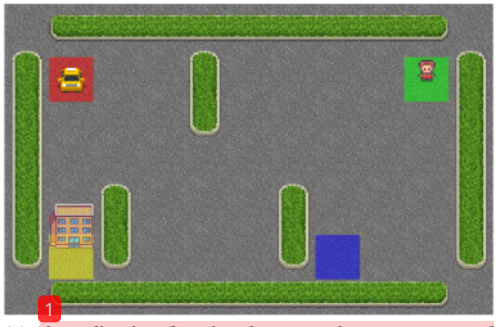
- Deterministic policies:
  Deterministic policies specify a unique action for each state.
- Stochastic policies:
  Stochastic policies specify a probability distribution over the set of actions for each state.



(a) (The policy for steering wheel control is a function that maps the current state of the vehicle and the environment to an action, which is the angle of the steering wheel)



(b) (The policy is a function that maps the current state of the system to an action, which is the force applied to the cart.)



(c) (the policy is a function that maps the current state of the system to an action, which is the next move that the taxi should take)

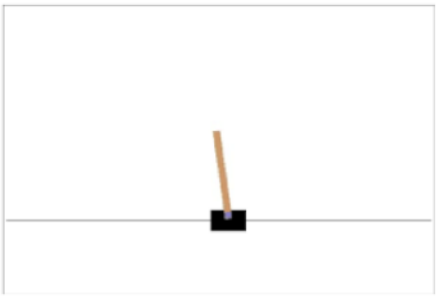
(d) (The policy is a function that maps the current state of the system to an action, which is the next move that the agent should take)

Figure 6: The figure illustrates the four different policies

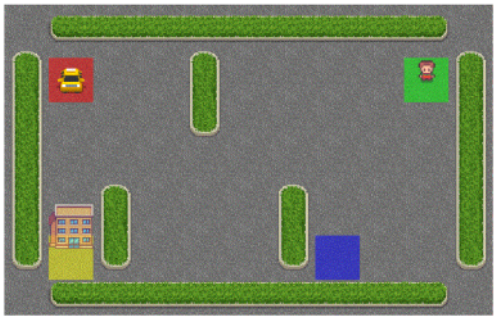
16. Exploration: *The Exploration* is the process of trying out various activities to gather data and understand the surroundings. This is crucial because it enables the agent to figure out which behaviours yield the greatest rewards and to develop a deeper comprehension of the world.

17. Exploitation: *The Exploitation* is the process of maximizing the payoff by utilizing the knowledge that the agent has already acquired. Based on the knowledge the agent has gained through exploration, this entails

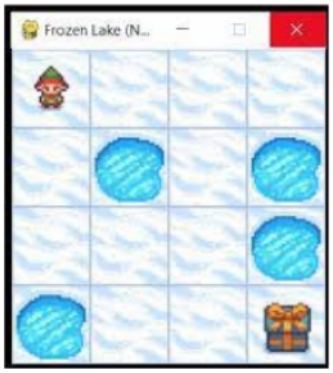
(a) (exploration refers to the process of trying out different steering wheel control actions (e.g. turning left, turning right, maintaining the current steering angle) in different driving scenarios (e.g. different road conditions, traffic conditions) to gather information about the environment and learn the best steering wheel control actions to take in each scenario)



(b) (refers to the process of trying out different actions (e.g. applying a force to the left or to the right) to gather information about the environment and learn the best actions to take in each state. For example, the agent may explore applying different forces to the pole in order to learn how to balance it)



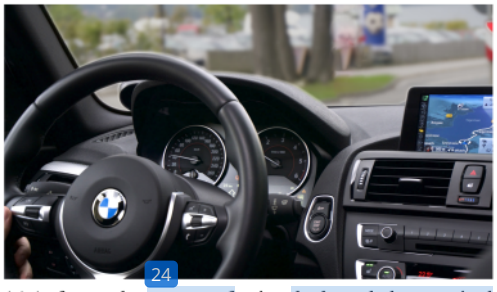
(c) (the agent needs to explore different routes to pick up and drop off the passenger to learn which route is the shortest. The agent also needs to explore different actions such as pick up, drop off, move up, down, left or right to learn which action is the best for each state)



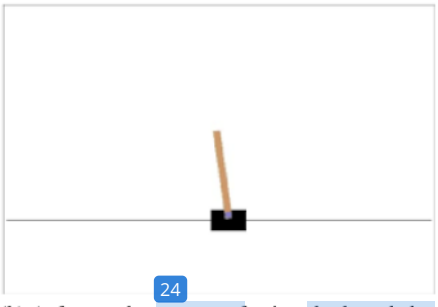
(d) (refers to the process of trying out different actions (e.g. moving in different directions) to gather information about the environment and learn the best actions to take in each state. For example, the agent may explore moving in different directions to find the correct path to the goal)

Figure 7: The figure illustrates the four different explorations

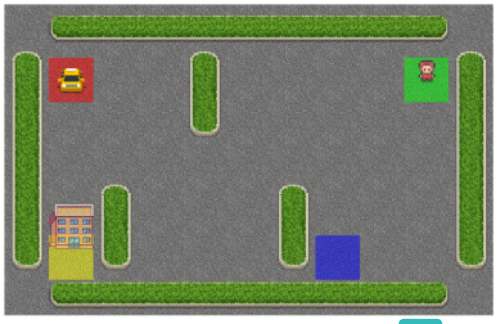
choosing the activities that are most likely to result in the highest reward.



(a) (refers to the process of using the knowledge acquired from the exploration to take the best steering wheel control action in a given driving scenario. For example, if the car has learned that the best steering angle on a winding road is to turn slightly left, the car should exploit this knowledge by turning slightly left every time it encounters a winding road)



(b) (refers to the process of using the knowledge acquired from the exploration to take the best action in a given state. For example, if the agent has learned that applying a small force to the right is the best action to take when the pole is slightly tilted to the left, it should exploit this knowledge by applying a small force to the right every time it encounters a similar state)



(c) (if the agent has learned that the shortest route to pick up and drop off the passenger is from location A to B and then from B to C, the agent should exploit this knowledge by taking the shortest route in each time it has to pick up and drop off the passenger)

(d) (refers to the process of using the knowledge acquired from the exploration to take the best action in a given state. For example, if the agent has learned that moving right is the best action to take when it's in a certain state, it should exploit this knowledge by moving right every time it encounters a similar state)

Figure 8: The figure illustrates the four different exploitations

18. Markov Decision Process (MDP): *An Markov Decision Process (MDP)* is a framework for representing decision-making in circumstances where the outcome is influenced by a series of actions and the system's current state.

    A mathematical framework called a Markov Decision Process (MDP) models how an agent communicates with the payment environment in the context of a payment environment. The MDP consists of a set of states, actions, rewards, and transition probabilities. The current context or information that the agent has at any one time, such as the transaction charges, the user, the merchant, processing time and the accessible payment types,

can be referred to as the system's state in the payment environment. A vector of pertinent attributes, such as the success rate, processing time, and transaction fees of each accessible gateway, can be used to represent the state. The agent has the option of choosing a specific payment type, retrying a transaction, or cancelling a transaction in the payment environment. Costs for each action may include transaction fees or fines for failing transactions. The rewards in the payment environment can be defined based on the outcome of the transaction, such as the success rate of the selected gateway, the processing time of the selected gateway, and any transaction charges incurred. The rewards can be positive or negative, depending on the outcome of the transaction and any associated costs. The possibility of changing states based on the agent's actions is represented by the transition probabilities in the payment environment. On the basis of previous data, these probabilities can be calculated, or they can be discovered by interacting with the payment environment. In conclusion, a set of states, actions, rewards, and transition probabilities make up the MDP for a payment environment. The state represents the current context or information that the agent has access to, the actions the potential decisions the agent might make, the rewards the results of the transactions, and the transition probabilities the likelihood that the agent will change states as a result of their actions. The MDP can be utilized to model the relationship between the agent and the payment environment and to identify the appropriate payment gateway selection strategy for a certain transaction.

Finding a policy—a rule or collection of rules that the payment environment can adhere to when making decisions and attempting to achieve a specific reward or objective—is the aim of the MDP. For instance, the success rate, transaction fees, and processing time all affect the payout. The MDP enables the payment environment to navigate the environment and accomplish its goals in a regulated and effective manner by allowing it to make decisions based on its current state and the anticipated results of certain actions.

19. Value Function or State Value Function: *The Value Function* The Value Function calculates the anticipated future benefit of a particular situation. It helps the agent make decisions and decides what the best course of action is in each state. It assesses the state's goodness by mapping the state to its value. The value function is a crucial component of the value iteration method and is used to assess the quality of the policy.
The value function is designated as V(s) for a given state s, under a policy of π, and it indicates the anticipated total of future rewards that the agent will earn starting from state s at time t and thereafter, according to the policy.

$$V\pi(s) = E\pi[G_T|S(T) = s]$$

A good value function, The value function V*(s) relates to the best course of action, which is the course of action that maximizes expected reward over time. The state value function is another name for it. It makes an assessment of a state's worth. It reveals the quality of a state's value in a given state.

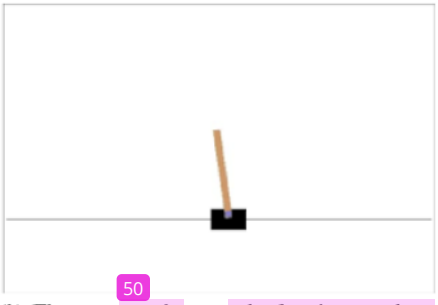
The expected return that an agent can obtain by adhering to a particular policy can be described as the value function in the context of a payment system. The value function can be described more precisely as the total of discounted future benefits that an agent can anticipate receiving, starting from a specific condition and adhering to a particular policy. The value function for a payment environment can be stated mathematically as follows:

$$V(s) = E[R|s, \pi]$$

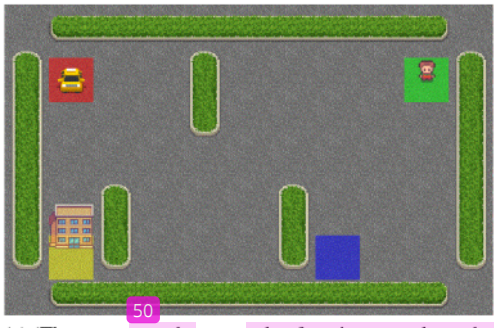
where s is the current state of the payment environment, π is the policy being followed by the agent, and R is the cumulative reward or return that the agent can expect to receive by following the policy π starting from state s. Value iteration, also known as policy iteration, is a method for estimating the value function in which the agent iteratively revises its estimate of the value function in light of its experience with the payment environment. The agent can utilize the estimated value function to decide which course of action will maximize the predicted future reward in each state. By choosing a particular payment gateway for a given transaction, an agent might anticipate earning a certain return, which can be estimated using the value function. The value function can be used to identify the best strategy for choosing the best payment type for a given transaction by taking into account the success rate, processing time, and transaction fees of each accessible gateway.

20. Q Value Function or Action Value Function: *The Q Value function* calculates the predicted future benefit of carrying out a specific activity in a particular condition. The Q-value function describes the expected return that an agent can obtain by carrying out a specific action in a specific state and then adhering to a specific

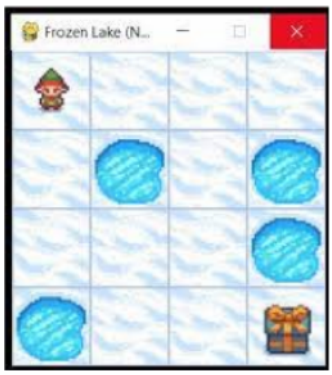
(a) (if the car is in a state where it is approaching a busy intersection, the value function would assign a high value to the state, indicating that it is important to be cautious in that situation. On the other hand, if the car is in a state where it is driving on a deserted road, the value function would assign a lower value to the state, indicating that it is safe to drive faster)



(b) (The agent uses the state value function to evaluate the potential benefits of being in different states)



(c) (The agent uses the state value function to evaluate the potential benefits of being in different states)



(d) (The agent uses the value function to evaluate the potential benefits of being in different states)

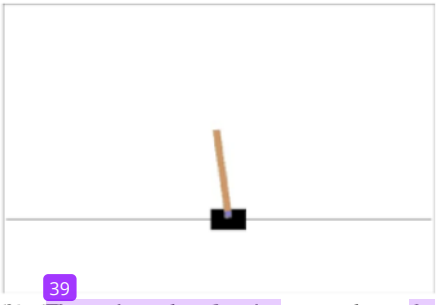
Figure 9: The figure illustrates the four different state values

policy going forward, additionally referred to as the action-value function when referring to a payment system. The Q-value function can be used to determine the best course of action to pursue at each step by picking the option that maximizes the predicted future benefit. The mathematical formulation of the Q-value function for a payment environment is as follows:

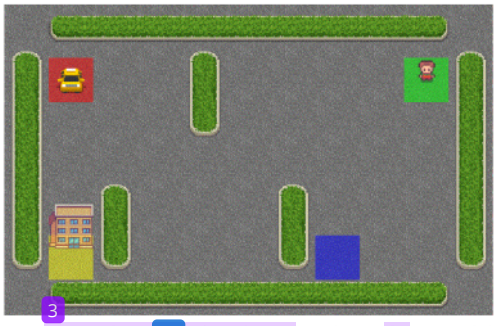
$$Q(s, a) = E[R|s, a, \pi]$$

where s represents the current state of the payment environment, a represents the action being taken by the agent, π represents the policy being followed by the agent, and R represents the cumulative reward the agent can anticipate receiving by adhering to the policy π starting from state s and taking action a. Through a procedure known as Q-learning or Sarsa, the agent can estimate the Q-value function by iteratively revising it based on its experience in the payment environment. The agent can utilize the estimated Q-value function to choose the course of action that will maximize the predicted future reward in each state. The Q-value function can be used in the context of a payment environment to calculate the expected return that an agent can obtain by choosing a particular payment gateway for a given transaction. The Q-value function can be used to choose the best gateway for a particular transaction by taking into account the success rate, processing time, and transaction fees of each accessible gateway. The state Value function, which is defined on a specific state, reveals the caliber of that state. But an action that is performed on that state is what is referred to as a "action value." It explains the merits of a specific action value in a particular situation. The Q-value function balances

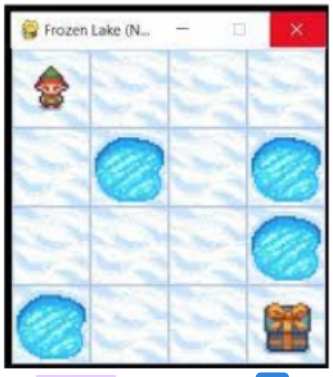
(a) (if the car is in a state where it is approaching a busy intersection, the action value function would assign a high value to the action of slowing down and a low value to the action of accelerating. Similarly, if the car is in a state where it is driving on a deserted road, the action value function would assign a high value to the action of accelerating and a low value to the action of slowing down)



(b) (The action-value function to evaluate the potential benefits of taking different actions in different states)



(c) (The action-value function to evaluate the potential benefits of taking different actions in different states)



(d) (the action-value function to evaluate the potential benefits of taking different actions in different states)

Figure 10: The figure illustrates the four different action values

the weighting of immediate rewards vs future rewards using the discount factor γ. Long-term benefits would be given priority if γ had a high value, whereas short-term prizes would be given priority if γ had a low value.

$$Q\pi(s, a) = E[G|s, a, \pi]$$

Model: *The Model* is a mathematical representation of the surroundings in which an agent operates. It helps agents learn about their environments and make wise judgments by forecasting the effects of their actions. It is possible to think of a model as a function that accepts an action and a state as inputs and outputs the next state and a reward. Without having to carry out such activities in the environment, the model enables the agent to simulate the impact of various acts in various states. In situations where the environment is expensive or time-consuming, this enables the agent to learn about it without having to interact with it directly. The model, which the agent may or may not be aware of, determines how the environment responds to particular behaviours. This distinguishes between the two situations.

- model-based: Model-based agents are those that are aware of the model. If we are aware of the environment, we can use dynamic programming to identify the optimal solutions.
- model-free: In a model-free environment, the agent attempts to explicitly learn the model as part of the algorithm and makes judgments with insufficient knowledge since it is unaware of the model.

Depending on whether the next state and reward are entirely decided by the current state and action, or whether there is some element of randomness, models can be either deterministic or stochastic.

- Deterministic: In a specific state, deterministic agents always take the same action.
- Stochastic: Stochastic agents select a course of action based on a probability distribution.

The model can be categorized based on value and policy.

- Value-based: The value function that calculates the predicted future reward for each state or state-action pair is learned by value-based agents.
- Policy-based: Agents that are policy-based learn a policy directly.

21. Markov Property:
The concept of the *Markov Property* says that, given a system's current state, its future state is independent of its prior states. As a result, only the current condition or state and the options for taking action define the likelihood of changing to any certain future state.
Let S be a set of states, A be a set of actions, and T be a transition that determines the likelihood of changing from one state to another based on the current state and action. This is how the Markov property is expressed. The following requirement is met by the transaction function if a system possesses the Markov property.

$$T(s'|s, a) = T(s'|s', a')$$

The probability of changing to state s' from state s, given action a, is equal to the probability of changing to state s' from state s', given action a', according to this equation. According to this equation, transition probabilities, which define the possibility of changing states, are dependent on the current state and activity. The idea that the present condition is unrelated to the prior states is not implied. In reality, because the transition probabilities take the system's history trajectory into account, the past states can have an impact on the future state.
The Markov property is a crucial presumption in RL since it enables the agent to describe the environment as a Markov decision process (MDP), which can be used to optimize expected reward while making decisions and taking actions.

22. State Transition Matrix:
The State Transition Matrix is a mathematical illustration of the likelihood that a state will change based on the current state and action. The state transition matrix T is a matrix where each element T[i, j] denotes the likelihood that, given a specific action a, state i will change to state j. Typically, the matrix is defined as follows:

$$T[i, j] = P(s' = j|s = i, a)$$

where s' is the next state, s is the current state, and a is the action taken. A key idea in RL is the state transition matrix, which enables the agent to describe its surroundings as a Markov Decision Process (MDP) and choose decisions that maximize the predicted reward. The Bellman equation or other algorithms can be used to determine the expected reward for each state and action using the matrix. The agent can then choose the activities that are most likely to result in the greatest reward by using this information to direct their decision-making process.

23. Process: In Reinforcement terminology, A *process* is the progression of an agent's behaviors across time and the conditions of their surroundings.

24. Discrete Time Process: A discrete-time process in reinforcement learning (RL) is a series of states, behaviors, and rewards that develop over discrete time steps as opposed to continuously. In a discrete-time process, the agent can act at regular intervals, and these regular intervals also govern how the environment will react to the agent's activities. Because they allow for straightforward and effective algorithms, discrete-time processes are frequently employed in RL; nevertheless, they can also be less realistic than continuous-time processes.

A discrete-time process is a series of time steps in which an agent interacts with the environment, for as in a payment environment. The agent monitors the state of the payment environment at each time step, acts in response to the observed state, and is rewarded as a result of the action made and the resultant state change. A Markov Decision Process (MDP), which is composed of a set of states, a set of actions, a reward function, and a state transition function, can be used to formalize the discrete-time process for a payment environment. The agent observes the state of the payment environment at each time step, chooses an action based on the state that is being observed, and is rewarded based on the action chosen and the state transition that results from it. Given the present state and the action taken, the state transition function gives the

probability distribution over the subsequent state. A discrete-time process can be used to simulate the process of choosing the best payment gateway for a specific transaction in the context of a payment environment. The various payment gateways, their associated success rates, processing times, and transaction fees, as well as the transaction amount, are among the details the agent can examine at each time step of the payment environment. The next step for the agent would be to choose a payment gateway for the transaction. The agent receives a compensation based on the chosen payment gateway and the outcome of the transaction; this reward may be positive or negative depending on the success rate, processing time, and transaction fees of the chosen payment gateway. Given the current state and the action taken, the state transition function provides the probability distribution over the following state. This probability distribution may vary depending on the payment gateway's success rate, processing time, and transaction fees.

25. Stochastic Process: A series of random variables that change over time is referred to as a stochastic process. A stochastic process is a collection of random variables that, in the context of a payment environment, describe the environment's state at each time step. It is assumed that these random variables follow a probability distribution, which may be affected by the agent's activities. As a Markov Decision Process (MDP), which is composed of a set of states, a set of actions, a reward function, and a state transition function, the stochastic process for a payment environment can be described. The agent monitors the state of the payment environment at each time step, chooses an action based on the observed state, and is rewarded based on the action chosen and the subsequent state transition. The state transition function, which assumes stochastic behavior, describes the probability distribution across the next state given the present state and the action taken. The stochastic process can be used to simulate the uncertainty involved in choosing a payment gateway for a specific transaction in the context of a payment environment. Information about the transaction amount, the payment gateways that are available, and their associated success rates, processing times, and transaction fees may all be included in the environment's state at each time step. The next step for the agent would be to choose a payment gateway for the transaction.Because the success rate, processing time, and transaction fees of the chosen payment gateway may fluctuate depending on outside factors like network congestion or server outages, the ensuing state transition may be stochastic in nature. Depending on elements like the success rate, processing time, and transaction fees of the chosen payment gateway, the agent receives a reward based on the state change that results. This reward may be positive or negative.

26. Discrete Time Stochastic Process: A series of random variables that indicate the environment's condition at each discrete time step might be referred to as a discrete-time stochastic process for the payment environment. An action is chosen based on the observed state of the payment environment at each time step, and a reward is given based on the action taken and the subsequent state transition.Given the present state and the action conducted, which is presumed to be stochastic in nature, the state transition function gives the probability distribution across the subsequent state. The discrete-time stochastic process can be used to simulate the uncertainty involved in choosing a payment gateway for a specific transaction in the context of a payment environment. Information about the transaction amount, the payment gateways that are available, and their associated success rates, processing times, and transaction fees may all be included in the environment's state at each time step. The next step for the agent would be to choose a payment gateway for the transaction. Because the success rate, processing time, and transaction fees of the chosen payment gateway may fluctuate depending on outside factors like network congestion or server outages, the ensuing state transition may be stochastic in nature. Depending on elements like the success rate, processing time, and transaction fees of the chosen payment gateway, the agent receives a reward based on the state change that results. This reward may be positive or negative. In general, a discrete-time stochastic process for the payment environment can be a helpful tool for modeling and improving payment systems because it enables agents to consider the inherent uncertainty connected to payment transactions and choose payment gateways that minimize processing times and transaction fees while maximizing success rates.

    A series of random states occurring at distinct time increments is how a discrete time stochastic process is visualized.
    $S0, S1, S2, S3 \ldots St.$

27. Discrete Time Markov Process: *A discrete-time Markov process* (DTMP) is a kind of discrete-time stochastic process in which the system's future state depends only on its present state and not on any previous states. This means that it is a series of random variables that evolve across discrete time steps, and the chance of being in a specific state at a given time step solely depends on the state at the previous time step.

    A Discrete Time Stochastic Process is called Markov Process when the future state S_{T+1} depends solely on the current state S_T given the whole history of states.

$$P[S_{T+1}|S0, S1, S2, S3...S_T] = P[S_{T+1}|S_T]$$

The state of the system changes across discrete time steps with the Markov property in a discrete-time Markov process for the payment environment. This stochastic process ensures that the system's future state depends only on the present state and not on any previous states. The dynamics of choosing a payment gateway for a given transaction can be modeled using a discrete-time Markov process, where the state of the system at each time step includes details such as the transaction amount, the payment gateway types that are available, and their associated success rates, processing times, and transaction fees. Given the current state and the action that is received, which is presumed to be stochastic in nature, the state transition function gives the probability distribution across the subsequent state. The ability to quickly calculate the expected value of the return, or total reward, that an agent can anticipate receiving over a specific time horizon, is one of the main benefits of utilizing a Markov process to represent the payment environment. The Bellman equation, which sums the immediate reward earned at any time step with the expected return obtained from the following state, can be used to calculate this. Overall, a discrete-time Markov process for the payment environment can be an effective modeling and optimization tool for payment systems because it enables agents to consider the inherent uncertainty of payment transactions and choose payment gateways that reduce processing times and fees while maximizing success rates, all while quickly computing the expected return.

Given the present state of the system and the agent's activities, a DTMP is an effective mathematical tool in RL. It enables the agent to predict the future states of the system.It helps to model the problem as a Markov Decision Process (MDP), which is the cornerstone of RL, is helpful. MDPs are used to represent decision-making issues when the agent's actions have an impact on the system's probabilistic evolution over time.

28. Markov Process: *The Markov Process* or The stochastic process known as the Markov Chain satisfies the Markov property and is named after the Russian mathematician Andrey Markov. Accordingly, the likelihood of a future state depends just on the present state and the passage of time, rather than the series of events that came before it. A Markov process is a tuple (s, p). S is a finite set of states ($s \in S$). P is a state transition probability matrix.

$$PSS' = P[S_{T+1}|s_T = s]$$

In Markov Process there is no action and no rewards.A state space, which is the collection of all potential states that a Markov process may occupy, and a transition probability matrix, which describes the probabilities of changing states, can be used to describe a Markov process.

29. Markov Reward Process:
The Markov Reward Process is a mathematical framework used to explain how an agent would behave over the long term in a specific environment. The system consists of a set of possible states, actions, and rewards for the agent as well as a set of probabilistic rules that specify how the agent changes states and receives rewards.

A Markov Reward Process is Markov Process with rewards. (MRP = MP + R) An MRP is defined by a tuple (S, P, R, γ), where:

S is a finite set of states that the agent can be in. ($s \in S$)
P is a set of transition probabilities that define the probability of transitioning from one state to another given the current state and action taken.
R is a reward function that defines the reward received by the agent when transitioning from one state to another.

$$R_S = E[R_{T+1}|S_T = s]$$

In MRP(Markov Reward Process, there is no action. γ is a discount factor that dtermines the value of potential benefits against current ones. Higher numbers suggest a stronger emphasis on long-term rewards, and the scale ranges from 0 to 1. In an MRP, the behavior of the agent is modeled as a series of states and actions, with the agent acting and changing states in accordance with probabilities set forth by P and earning rewards in accordance with the function R. The expected return, also referred to as the expected total of discounted future benefits, is what the agent seeks to maximize. In addition to serving as the foundation for many RL algorithms, MRPs are a powerful tool for modeling and understanding RL issues. They can be used to identify and address issues like determining the best course of action for an agent to take in a particular environment or calculating the worth or value of a specific state or action.

A MRP for the payment environment can be more fully described by the elements listed below: the potential configurations of the payment environment represented by a collection of states S, including the payment gateways that are accessible, their success rates, processing times, and transaction fees. Given an action by the agent, a transition probability function P specifies the likelihood of a transfer from one state to another. The Markov property of this function is satisfied, which means that the likelihood of changing from one state to another in the future depends only on the current state and the action taken, not on any previous states. a reward function R that gives each state of the system a numerical value to represent the immediate reward the agent receives when it enters that state. The incentive function in a payment environment can take into account things like the fees, success rates, and processing times connected to each payment gateway, as well as fines for failed transactions or other negative outcomes. When calculating the expected return, a discount factor called γ balances the relevance of present benefits with that of future rewards. An MRP for the payment environment can be used to calculate the expected return or total reward that an agent can anticipate receiving over a specified time horizon under a specific policy using these components. The Bellman equation, which represents the expected return at any time step as the sum of the immediate reward at that time step and the expected return from the following state, discounted by the discount factor, can be used to perform this calculation. In conclusion, an MRP can be a useful tool for modeling and optimizing payment systems because it enables agents to consider the inherent uncertainty and reward structure linked to payment transactions, choose payment gateways that reduce processing times and transaction fees while boosting success rates, and compute expected returns effectively.

30. Markov Decision Process: *The Markov Decision Process* MDP is a mathematical model that defines the decision-making problem faced by the agent, must solve when deciding which payment gateway is best. An extension of MRP, a Markov Decision Process (MDP) adds the decision-making issue of an agent, allowing the agent to select one action from a list at each time step. The objective is to identify the best policy that maximizes the predicted cumulative reward over time. The agent's activities have an impact on how the system evolves. A set of states, a set of actions, a transition probability, and a reward function serve to define an MDP. As a result, MDP has all the components of MRP as well as the decision-making issue, where the agent must select the optimum course of action from a list of options in order to maximize the predicted cumulative reward over time.

A set of states, a set of actions, a transition function, and a reward function all go into defining an MDP. A Markov Decision Process (MDP) in the context of a payment environment can be described as a tuple. (S, A, P, R, γ), where:

> S is a collection of states that depicts the many configurations of the payment environment, including the payment gateways that are accessible, their success rates, transaction fees, and processing times.
> A is a set of options available to the agent, including choosing a certain payment gateway to complete a transaction.
> P is a function that calculates the likelihood that, given an action made by the agent, a state will change from one to another. The Markov property of this function is satisfied, which means that the likelihood of changing from one state to another in the future depends only on the current state and the action taken, not on any previous states.
> R is a reward function that gives each state of the system a numerical value to represent the immediate reward the agent receives when it enters that state. The incentive function in a payment environment can take into account things like the fees, success rates, and processing times connected to each payment gateway, as well as fines for failed transactions or other negative outcomes.
> γ is a discount factor used in the calculation of expected return to balance the weighting of current rewards vs future rewards.

The expected return or total reward that an agent can anticipate receiving over a specified time horizon can be maximized by using an MDP for the payment environment and these components. The Bellman optimality equation, which expresses the maximum expected return achievable from any state as the sum of all possible actions of the immediate reward obtained from that action and the expected return obtained from the next state, discounted by the discount factor, is used to determine the optimal policy, which is a function that maps each state in S to an action in A. All things considered, an MDP offers a strong framework for modelling and improving payment systems because it enables agents to consider the inherent uncertainty and reward structure linked to payment transactions, choose payment gateways that reduce processing times and transaction fees while maximizing success rates, and compute the best possible policy quickly.

2 Introduction

E-commerce and mobile technology have completely changed how we pay for goods and services in today's society. We now routinely make payments online and via mobile devices because they make handling our money easier, safer, and faster. With online payments, customers can make purchases whenever they want, from anywhere in the world, without having to go to a physical business or bank. Businesses now have more ways to reach and engage customers in new markets and across countries and provide more flexible payment alternatives as a result. Because smartphones and mobile apps are now widely used, the industry of mobile payments has grown significantly in recent years. Now more than ever, consumers can manage their finances while on the go thanks to the ability to make payments using their mobile devices, whether in-person, online, or peer-to-peer.

Due to their widespread use and ability to offer more ease, flexibility, and security in financial management, online and mobile payments have become an integral part of our daily lives.

Payment aggregators have been instrumental in making the process of collecting electronic payments for businesses more straightforward, which has helped to promote the rise of online and mobile payments. Businesses can simplify the payment process and give their consumers more payment options by collaborating with payment aggregators instead of having to maintain relationships with many payment processors. Due to this, online and mobile payments are now more widely used, making it simpler for companies of all sizes to engage in the digital economy.

The essential elements in the processing of electronic payments. Here is a quick synopsis of each:

1. Payment aggregator(s): An organisation that enables businesses to collect payments from clients electronically.

2. Customer: The person making the payment.

3. Issuing bank: The financial institution that issued the customer's chosen payment method.

4. Merchant: The business or entity that is receiving payment.

5. Acquiring bank: The financial institution handling the merchant's payment processing.

6. Processor: The business in charge of handling the payment's actual processing, which includes checking the payment method and transferring money from the issuing bank to the acquiring bank.

7. Payment gateway: The software that enables a transaction by linking the merchant's website or mobile app to the processor and acquiring bank.

8. Smart Agent: The Smart Agent seeks to choose the best gateway, one that increases the likelihood of success while minimising processing times and transaction costs. It develops its intelligence by using previous data and a trial-and-error method, rewarding successful outcomes and punishing bad ones.

Together, these elements make sure that payments are completed securely and quickly, enabling businesses to accept a variety of payment options and facilitating customers' payment processes.

The ecosystem of an e-commerce platform is depicted above, in which clients place orders on merchant websites via mobile or the internet, and payment processors function as middlemen, facilitating successful money transfers from customer accounts to merchant accounts. Among the components of this ecosystem for processing payments are payment gateways, networks, and aggregators.

Payment gateways serve as the primary points of contact between customers and payment networks in this ecosystem. Routing transactions to the relevant issuing banks for permission and back to the acquiring banks for settlement is the responsibility of payment networks. Payment aggregators collect and distribute services for payment processing from various payment networks and gateways.
With numerous parties engaged in the transaction process, such as customers, merchants, payment gateways, payment networks, acquiring banks, issuing banks, and payment aggregators, the payment processing ecosystem is extremely complicated. Every organisation is essential to the success of the payment transaction.

Numerous machine learning techniques, including fraud detection algorithms, payment gateway selection algorithms, and transaction processing time optimisation algorithms, can be used to optimise the payment processing ecosystem. Numerous machine learning methods, including supervised learning, unsupervised learning, deep learning and reinforcement learning can be used to implement these algorithms.

Overall, the e-commerce platform's payment processing ecosystem is a crucial part, and machine learning algorithms are essential for streamlining payment transactions and maintaining the platform's success.

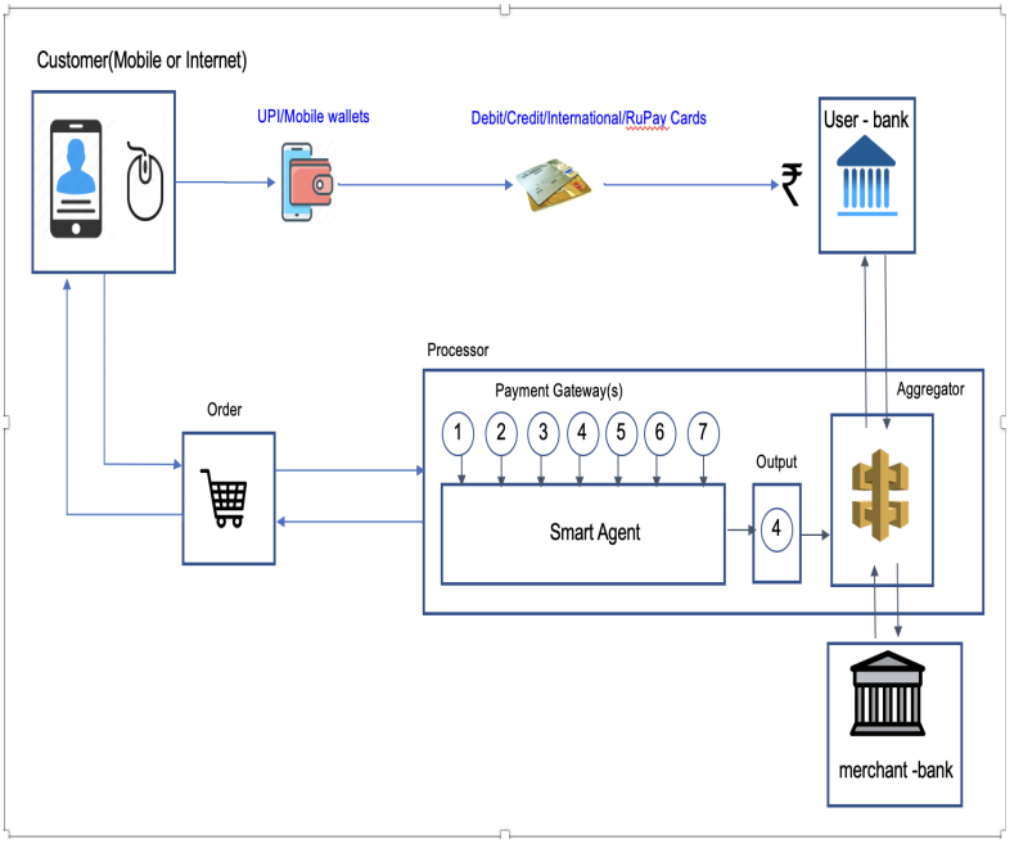


Figure 11: The Smart Agent selects the payment gateway that optimizes success rate while minimizing transaction charges and processing time, it the above gateway 4 has been selected by the smart agent.

In fact, payment aggregators are constantly coming up with new ideas to enhance the ecosystem of payment processing and give their clients the best possible service. They choose the best payment gateway for each transaction by using data analytics and machine learning techniques, for example.

Payment aggregators can spot patterns and trends in payment success rates and processing times for each payment gateway by looking at past transaction data. As a result, they may create predictive models that calculate the chances of a successful transaction with each gateway for a certain transaction, taking into account elements like the transaction's value, location, and payment type.

Payment aggregators can dynamically choose the payment gateway that will result in a transaction that is most likely to be successful while also minimising processing times and transaction fees based on these predictions. Decreasing the possibility of failed transactions or delays in payment processing enhances both the overall success rate of transactions on their platform and the consumer experience.

3 Literature Survey

3.1 Payment Eco Systems

Machine learning and data analytics are becoming increasingly important tools in the payment processing ecosystem, helping payment aggregators to optimize their operations, increase efficiency, and provide better services to their customers. The paper [1] outlines how they have used machine learning algorithms to optimize their payment routing system and improve the success rate of transactions. The solution leverages historical transaction data to build predictive

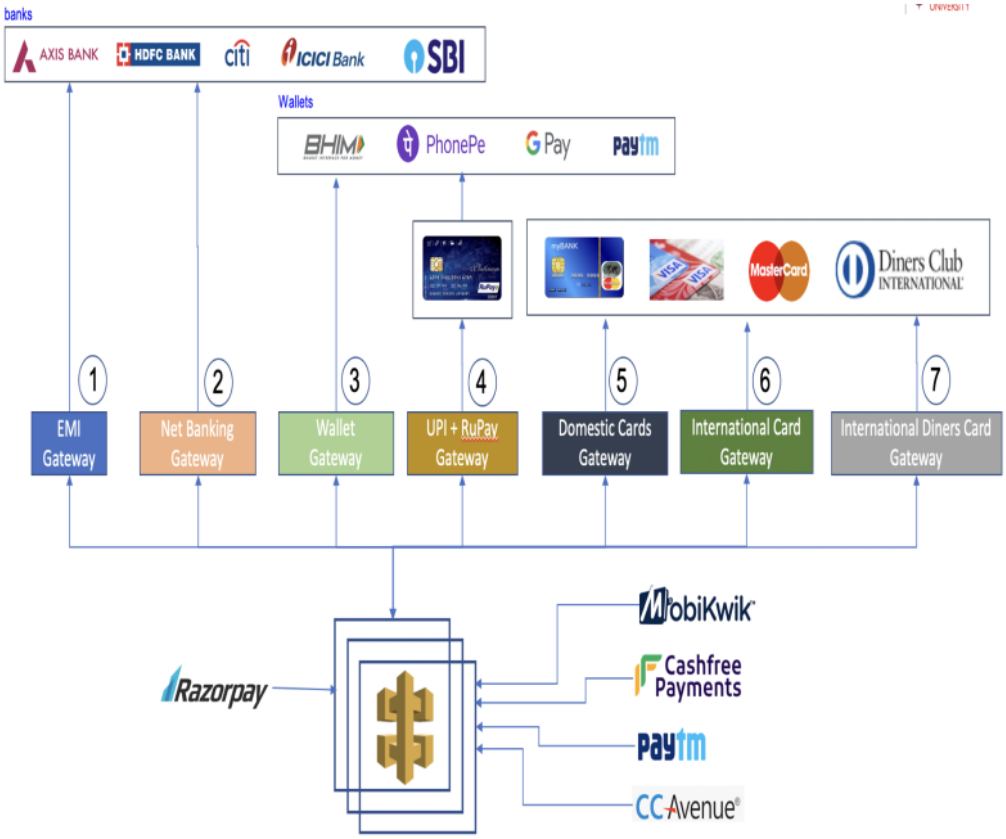


Figure 12: The Smart Agent selects the payment gateway that optimizes success rate while minimizing transaction charges and processing time.

models that estimate the likelihood of a successful transaction for each payment gateway. These models take into account various factors such as transaction amount, success rate, transaction charges, processing time and status to predict the optimal payment type for a given transaction. The paper also highlights how they have addressed the challenges of dealing with high-dimensional, noisy, and imbalanced data, which is common in the payment processing industry. They have used techniques such as feature selection, feature engineering, and outlier detection to ensure that the predictive models are accurate and reliable. Overall, Razorpay's AI-powered Smart Routing Solution is a great example of how machine learning can be used to optimize payment processing systems and improve the customer experience. With the increasing trend of e-commerce and online transactions, payment gateways have become an essential part of the digital economy. They provide a secure and reliable platform for customers and merchants to process financial transactions, and enable businesses to expand 76r customer base globally by accepting payments from multiple payment methods [2]. most commonly supported payment methods include credit and debit cards, net banking, digital wallets, and UPI (Unified Payments Interface). By offering a variety of payment methods, payment aggregators can cater to a broader range of customers and ensure that they are able to complete transactions using their preferred payment methods. Additionally, supporting multiple payment methods can help to improve the overall success rate of transactions, as customers are more likely to complete a transaction if they are able to use a payment method that they are familiar with and trust. The popularity of payment aggregators can vary depending on the region, as some aggregators may be more popular in certain areas due to factors such as their marketing efforts, partnerships with local businesses, or the availability of specific payment methods that are popular in that region. It's important for payment aggregators to understand these regional differences and tailor their services accordingly to meet the needs of their customers in each region. To select the optimal payment type for a given payment aggregator, we can utilize data analytics and machine learning techniques. We can analyze historical transaction data and identify patterns and

Per Transaction Charges	Razorpay	CCAvenue	Cashfree Payments	paytm	MobiKwik
UPI/RuPay with Debit Card	2%	0%	0%	0%	0%
Net Banking	2%	2%	1.75%	1.99%	1.9%
Wallet	2%	2%	1.9%	1.99%	1.9%
Domestic Debit/Credit Cards	2%	2%	1.75%	1.99%	1.9%
EMI with Credit Card	2%	2%	1.5%	2.99%	2.25%
International	3%	3%	3.5%	2.99%	2.9%
International - Diners/Amex Cards	3%	4%	2.95%	2.99%	2.9%

Figure 13: The transaction charges are shown for each payment aggregator and the associated payment types for reference. This enables us to track how transaction fees vary among various aggregators.

trends in payment success rates and processing times for each payment type. Based on this analysis, we can develop predictive models that estimate the likelihood of a successful transaction with each payment type for a given transaction, taking into account factors such as the transaction amount, transaction fees, processing time, and status. This can help us to dynamically select the payment type that is most likely to result in a successful transaction while minimizing processing times and transaction fees. We can use various machine learning algorithms such as decision trees, random forests, logistic regression, or neural networks to build these predictive models. Once we have these models in place, we can continually improve them by incorporating new transaction data and refining our algorithms over time. Overall, by utilizing data analytics and machine learning techniques, we can optimize payment processing and increase the success rate of transactions while minimizing transaction charges and processing time. Using reinforcement learning algorithms can potentially improve the accuracy and efficiency of payment processing by taking into account complex and dynamic factors that may not be captured by traditional machine learning models. RL algorithms are designed to learn optimal policies through trial-and-error interactions with an environment, which can be useful in situations where the optimal policy may change over time or where there is a high degree of uncertainty or variability in the data. By using RL algorithms to select the optimal payment gateway for each transaction, payment aggregators can potentially have one or more objectives like improving the success rate of transactions, reducing transaction fees and processing times and providing a better experience for their customers.[3] One of the core objectives is to Minimise transaction fees is an important objective for payment aggregators, as it can help them to attract more customers and increase their revenue. By selecting the payment gateway with the lowest transaction fees for a given transaction, payment aggregators can reduce their own costs and pass on the savings to their customers, making their services more attractive and competitive in the market. Using RL algorithms to optimize payment gateway selection based on transaction fees can help payment aggregators to achieve this objective more effectively and efficiently than traditional

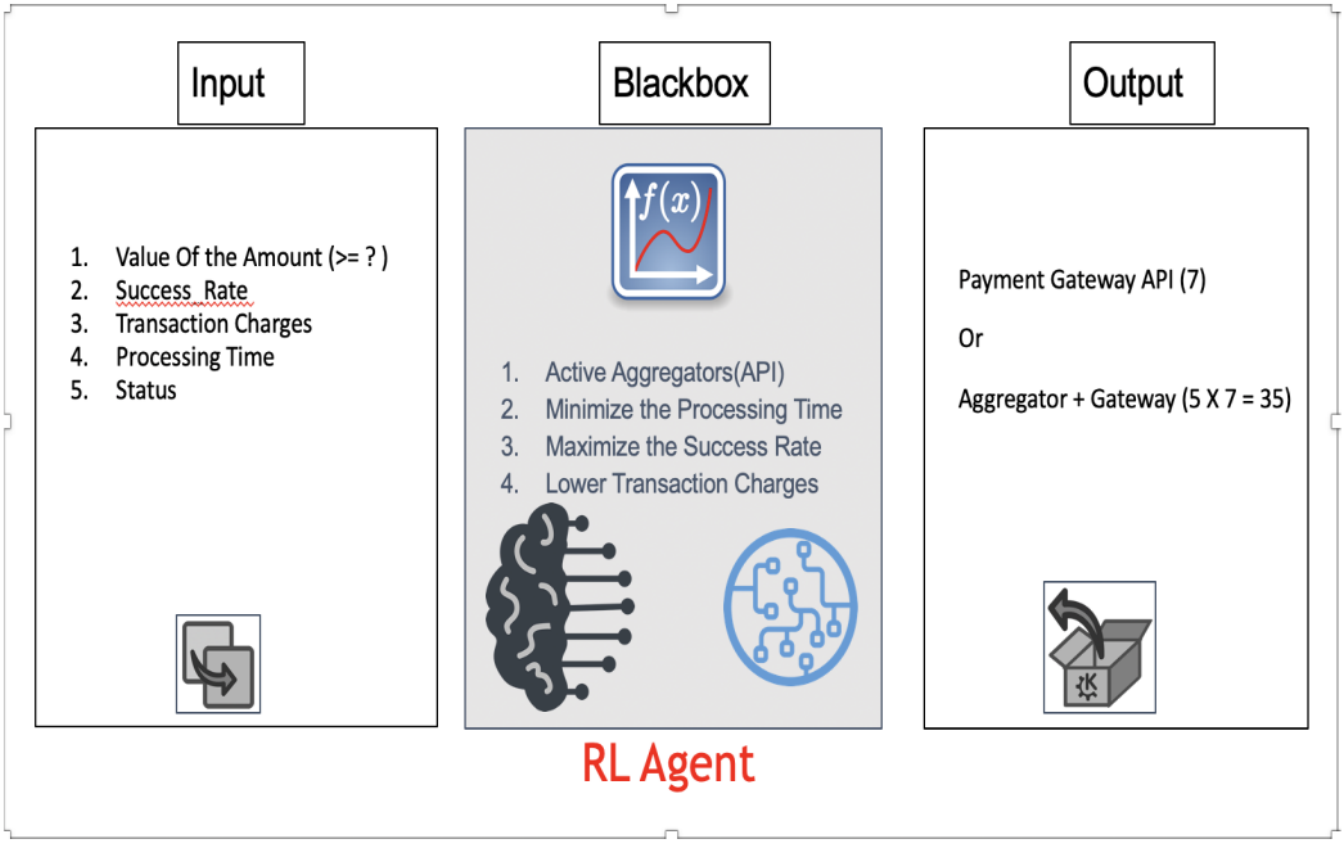


Figure 14: Reinforcement learning can learn optimal strategies and adapt to changing conditions. However, regression may not be able to capture complex interactions between features and neural networks can be computationally expensive and difficult to interpret

methods. Using RL algorithms to optimize payment gateway selection based on transaction fees can also lead to a more dynamic pricing model. By constantly analyzing transaction fees [4] and optimizing payment gateway selection, payment aggregators can adjust their pricing strategies in real-time to ensure that they are offering the most competitive rates to their customers. This can also help payment aggregators to stay ahead of the competition and attract new customers by offering lower transaction charges than their competitors.

4 Data Generation

The real-time data for a specific duration and payment data can be obtained by either engaging e-commerce service providers or creating real-time simulations through distributions.

1. **Amount**: The payment amount data is generated following the log-normal distribution. This distribution is typically used to model positive variables because it has a skewed distribution that closely resembles the distribution of payment amounts in the real world, where smaller values are more common and higher values are more unusual.
2. **Type**: The payment type is chosen at random from a predetermined list of payment kinds. In the payment ecosystem, this guarantees a diverse representation of various payment methods.
3. **Aggregator**: The aggregator is selected at random from a group of aggregators. A service provider known as an aggregator collects and manages payment transactions from numerous sources. Different aggregators are represented in the generated data thanks to random selection.

4. **Success Rate**: Each payment's success rate is calculated using the beta distribution. The success rates of payment transactions can be represented by this distribution, which allows modelling probabilities between 0 and 1. To account for variations in success rates among various payment methods, the alpha and beta parameters are set appropriately for each payment type.
5. **Processing Time**Each payment's processing time is calculated using a uniform distribution. With the help of this distribution, it is ensured that the processing time values fall within a continuous range of 0 to 10, simulating the time needed to complete various payment transactions.
6. **Transaction Charge**The chosen aggregator and payment type are combined to determine the transaction fee. A predefined dictionary is used to determine the precise transaction fees connected with each combination, ensuring accurate cost levels for various payment scenarios.
7. **Merchant Id** An integer value that is produced at random within a range is the merchant id. These numbers serve as distinctive identifiers for the businesses taking part in the payment operations.
8. **User Id**: An integer value that is produced at random within a range is the User Id. These numbers serve as distinctive identifiers for the businesses taking part in the payment operations.
9. **Status**: The generated success rate is used to determine each payment's status. The payment is classified as successful (status = 1) if the success rate is greater than a threshold of 0.60; otherwise, it is classified as failed (status = 0).

Overall, the use of these distributions and random picks guarantees that a variety of payment data is produced, capturing the variance and traits seen in actual payment transactions. The generated payment data is utilized to train and test an RL algorithm for choosing the best payment gateway based on several criteria, such as lowering transaction fees, increasing success rates, and speeding up processing.

Figure 15: Seaborn is a powerful data visualization library in Python. A pair plot is a way to visualize the relationship between multiple variables in a single plot. It shows pairwise relationships between variables in a dataset

In this example, df is a Pandas DataFrame that contains the payment data. We drop the amount, merchant_id, and user_id columns from the DataFrame because they are not useful for our pair plot. We set the diag_kind parameter to 'kde' to show a KDE plot at the diagonal. Finally, we set the hue parameter to 'status' to color-code the points based on whether the payment was successful or not.

5 Define Environment

5.1 Payment Environment

For the payment problem, the PaymentEnv class is defined by extending OpenAI Gym environment. The core methods description provided as shown below:

1. **Observation Space**: The five observation points that make up the environment are success rate, processing time, transaction cost, log amount, and status. These observations reveal details on the state of the environment.
2. **Initialization**: The init method is used to initialize the environment. This information represents the environment's starting point.
3. **Action Space**: A discrete action space with seven options is provided by the environment. At each phase, agents engaging with the environment can select one of these options.
4. **Step**: The environment provides a step method. This method returns the next state of the environment, the reward obtained from the action, and a flag indicating whether the environment has reached a terminal state.
5. **Reset**: The environment provides a reset method that returns the environment's current state to its initial state. The starting state is returned by this procedure.

5.2 Pre-processing

1. **Static Reward Calculation**: Calculation: The success rate, transaction fee, processing time, and status attributes are all combined in the computing the static reward for the historical data. The beta distribution is afterwards utilized to calculate the prior probabilities of the success rate of each payment type using the calculated static reward. The parameters (a and b) of the beta distribution, which indicate the prior belief about the success rate of each payment type, are calculated using the reward-adjusted counts (success count and failure count). The prior probability of success rates can be updated depending on the observed rewards by including the reward in the beta distribution, allowing the agent to learn and make wise decisions in the payment environment.
2. **Visualise Beta** distribution: We can depict various previous assumptions regarding the likelihood that an action will succeed using the beta distribution. For instance, if an action has a higher "a" value than "b," it denotes a greater prior conviction in the action's success. On the other hand, if b is greater than a, it denotes a stronger prior belief in the action's failure. We may see the prior probabilities for various success rates by showing the beta distribution for each action. The distribution's form sheds light on the uncertainty and unpredictability connected to the success rate of each action.

   In conclusion, the beta distribution in this case denotes the prior assumptions about the likelihood that each action will be successful, enabling probabilistic decision-making in the payment problem.

6 Problem Definition

Regression and classification are supervised learning techniques that can be used to make predictions based on labelled data. Reinforcement learning, on the other hand, is a type of unsupervised learning that is focused on learning how to make optimal decisions in an environment through trial and error. In many cases, problems that can be framed as reinforcement learning problems cannot be solved using regression or classification techniques because the optimal decision depends on the current state of the environment and the actions that have been taken in the past. In these cases, reinforcement learning can be a more suitable approach because it allows the agent to learn an optimal policy through trial-and-error interactions with the environment. Wrong selection of payment method may cause payment process disruptions for customers, resulting in lost revenue and inconvenience, and frustration for customers unable to make payments due to technical issues or being offline. There are THREE key points that justify framing the payment method selection as a reinforcement learning problem rather than a supervised machine learning problem:

1. Environment that is dynamic and ever-changing: As consumer preferences and transaction volumes evolve, so does the payment ecosystem. Because of this, using a static, pre-defined model in supervised machine

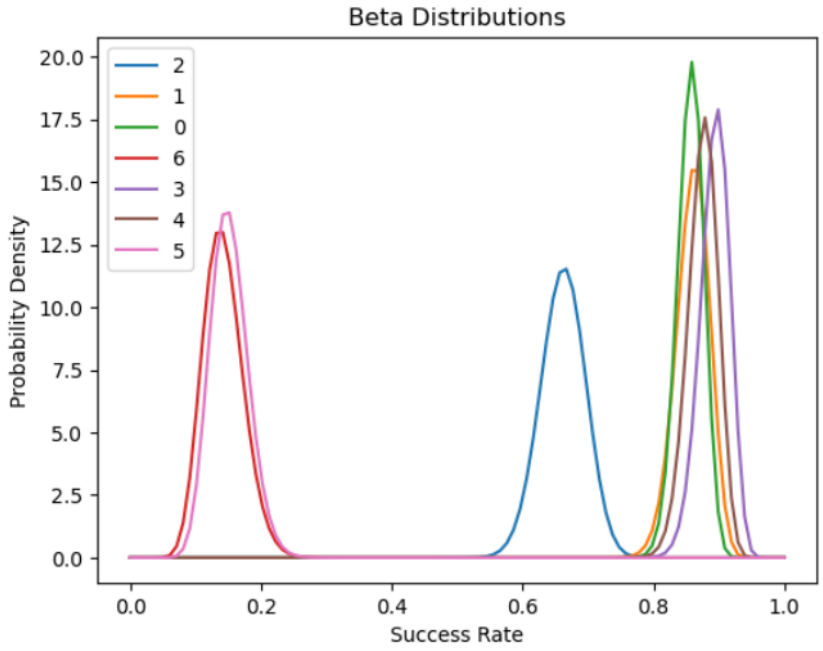


Figure 16: Beta distributions that depict the historical success rates for various payment actions. The distributions capture the prior assumptions regarding the likelihood of successful payments for each action, enabling the payment problem's probabilistic analysis and decision-making

learning is challenging. Because it can adjust to shifting circumstances over time, reinforcement learning is better suited to managing dynamic environments.

2. Long-term objectives: When choosing a payment method, the objective is to raise the success rate while lowering transaction fees and processing times in the long run. While reinforcement learning is better suited to long-term optimization, supervised machine learning is often focused on improving a single job.
3. Interactions between actions and results: When choosing a payment method, the method selected may have a big impact on the success rate and other results. While supervised machine learning might not be as effective in this situation, reinforcement learning is made to handle interactions between actions and results.

Overall, a reinforcement learning technique would be more appropriate given the payment ecosystem's complexity and dynamic nature, the long-term objectives of choosing a payment method, and the relationships between actions and results.

7 Define Learning Agents

7.1 Q-Learning Agent

The reinforcement learning technique known as Q-learning is used to resolve Markov decision processes (MDPs). It is an off-policy learning algorithm that discovers the best course of action by repeatedly improving a Q-function estimate. The predicted total reward of adopting a particular action in a particular state and then continuing the best course of action from that state on is represented by the Q-function, which is a value function. As part of the process, the environment is sampled repeatedly, the current state and the rewards for taking particular actions are observed, and the Q-function is then updated as necessary. The update rule comprises calculating the discrepancy between the Q-value's current estimate and a target Q-value based on the reward and the predicted value of the following state. The amount to which the current estimate is revised depends on the learning. Q-learning is a sort of reinforcement learning algorithm that determines the best action-value function Q(s, a) for a certain state-action pair (s, a) before

learning a policy. The Q-table is iteratively updated in Q-learning based on the observed rewards and the best Q-values for each state-action pair. The Q-table may be initially initialized with zeros or with arbitrary values. The Q-table would have 1000x7 dimensions in a discrete state and action space with 1000 state spaces and 7 action spaces(0 to 6). The Bellman equation, which integrates the observed reward and the anticipated ideal Q-value for the following state, is used to update the Q-values for each state-action combination during the learning process. The equation for Q-learning is as follows:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma * max(Q(s',a')) - Q(s,a)]$$

where:

s is the current state
a is the current action taken in state s
r is the observed reward for taking action a in state s
s' is the next state
a' is the action that maximizes the Q-value for the next state s'
α is the learning rate ($0 <= \alpha <= 1$) which controls the influence of new information on the Q-values
γ is the discount factor ($0 <= \gamma <= 1$) which controls the importance of future rewards

There are several iterations in the Q-learning algorithm. The agent observes the current state, chooses an action based on the Q-values in the Q-table, gets a reward, and then modifies the Q-values for the state-action pair after each iteration. Learning an ideal policy that maximizes the predicted cumulative reward over time is the ultimate goal of learning. The best course of action can be determined after the Q-table has been learned by choosing the course of action that maximizes the Q-value for a certain state. Since each state-action combination must have a value stored in the Q-table, Q-learning has the drawback of being unsuitable for problems with huge state spaces. Due to limitations in memory and computing, this is frequently impossible. Additionally, when learning in continuous state and action spaces, Q-learning may have sluggish convergence and instability. Additionally, it struggles with partial observability and delayed incentives.

7.2 DQN

DQN (Deep Q-Network) is a kind of Q-learning that approximates the Q-function using a deep neural network. As a result, the agent is able to acquire knowledge from high-dimensional and continuous state spaces. DQN employs target networks and experience replay to overcome Q-learning's instability. A Q-learning variation called Deep Q-Network (DQN) makes use of neural networks to make approximations of the Q-value function. By reducing the difference between the predicted Q-value and the target Q-value using a loss function, the agent in DQN learns to estimate the Q-values. Similar to Q-learning, this goal Q-value is determined using the Bellman equation. DQN does not require the usage of a Q-table to hold all of the state-action pairs in contrast to Q-learning. Instead, the Q-values for each state-action pair are approximated using a neural network. The current state is the neural network's input, and its output is a vector of Q-values for each action that might be taken. The agent interacts with the environment while learning and uses a optimization algorithms like Adaptive Moment Estimation (Adam), stochastic gradient descent (SGD) approach to update the neural network parameters. An epsilon-greedy strategy, which balances exploitation with exploration, is used by the agent to choose a course of action. A strong algorithm called DQN has been utilized to handle challenging issues involving high-dimensional state and action spaces. On a number of tasks, including playing Atari games and navigating in 3D settings, it has been demonstrated to perform better than conventional reinforcement learning methods. By utilizing a neural network to mimic the Q-function, the DQN overcomes the shortcomings of Q-learning in handling high-dimensional and continuous state spaces. By utilizing experience replay and target networks, it also overcomes the instability of Q-learning. In order to train the network, experience replay samples small batches of experiences and saves them in a replay buffer. As a result, learning is stabilized and the association between subsequent events is decreased. By adjusting the target Q-values during training, target networks are utilized to stabilize the Q-learning updates. The replay buffer's huge memory requirements and the computational difficulty of deep neural network training are among DQN's drawbacks. Additionally, DQN overestimates Q-values, especially when working with noisy or insufficient data. Finally, for DQN to work well, hyperparameters must be tuned carefully.

7.3 DDQN

The Deep Q-Network (DQN) algorithm is an extension, and its name is Double Deep Q-Network. One of DQN's main drawbacks, the overestimation of Q-values brought on by the target update's maximizing procedure, is addressed by DDQN. In order to get around this restriction, DDQN incorporates a second Q-network known as the target network, which is used to assess the Q-values of the subsequent state in order to decide on the subsequent course of action. As a result, the overestimation of Q-values is decreased, and the training process' stability is increased. The training

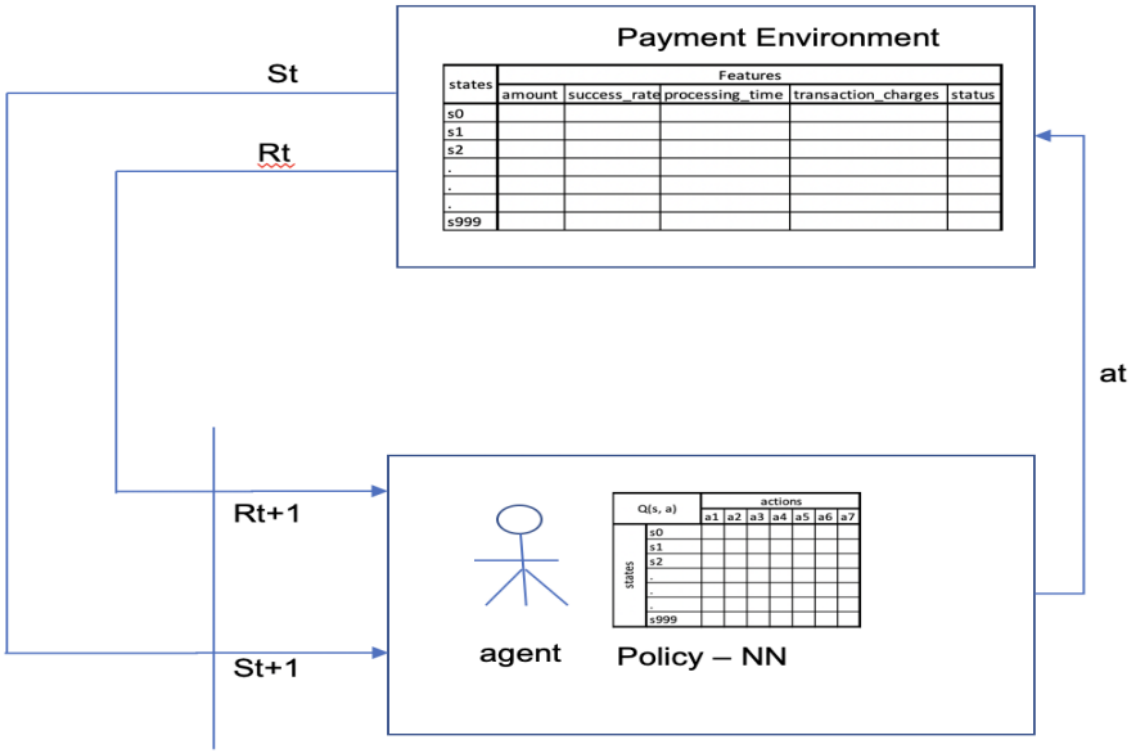


Figure 17: In Q-Learning, as the number of state spaces is discrete with 1000 and with 7 action spaces, the Q-Table is updated with optimal values for a given state with all the respective action spaces

process is stabilized in DDQN because the target network is changed less frequently than the Q-network. Every N episodes, where N is a hyper-parameter, the target network is updated with the weights of the Q-network. The provided code creates a class for the DDQN agent, initializes the hyper-parameters, uses Keras to build the Q-network and target network, stores experiences in a replay buffer, uses an epsilon-greedy policy to choose actions, trains the Q-network using samples from the replay buffer, updates the target network every N episodes, and runs the agent in the environment. A modification of the original DQN algorithm that solves the problem of overestimation of Q-values in some circumstances is the Double Deep Q-Network (DDQN). A "target network and a "online" network, two distinct neural networks, are used in DDQN. During the learning process, the target network is used to predict the ideal Q-values while the online network is utilized to choose actions. The update equation for DDQN is identical to that of DQN, but instead of selecting the best action from the target network and using the matching Q-value in the update equation, the maximum Q-value from the online network is utilized for the next state-action pair. As a result, the overestimation of Q-values that might happen when using the original DQN algorithm is reduced. In order to train an optimal policy that maximizes the predicted cumulative reward over time, DDQN and DQN share the same ultimate objective. However, DDQN is able to enhance the performance and stability of the learning process by utilizing two distinct networks. In conclusion, DDQN is a DQN algorithm version that employs two different neural networks to calculate the ideal Q-values and minimize overestimation. Similar to DQN, DDQN has constraints such the inability to handle continuous action spaces and the requirement for a significant amount of experience data in order to perform well. Furthermore, DDQN can still overestimate Q-values in some circumstances, such as in environments with few rewards or complicated action spaces.

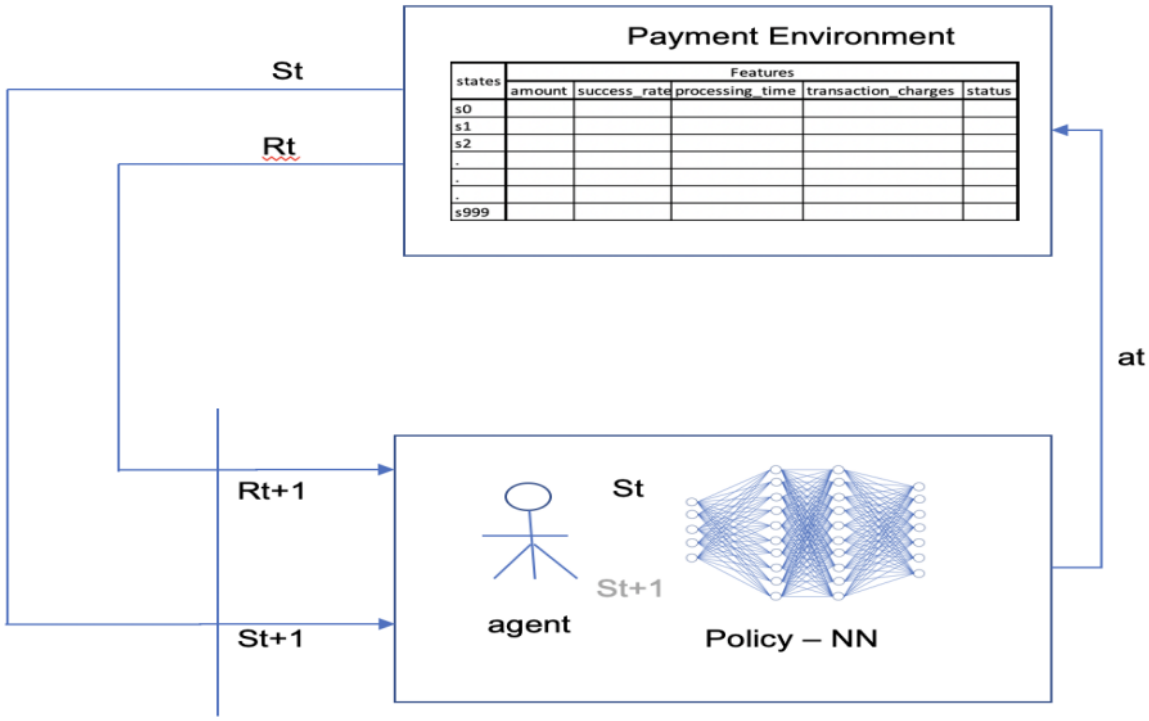


Figure 18: In DQN, a single network is used for both selecting actions and estimating their values. This network is trained by minimizing the difference between the predicted Q-values and the target Q-values, which are updated using a separate target network.

7.4 Dueling DDQN

Dueling architecture and Double DQN are combined to create Dueling Double Deep Q-Network (Dueling DDQN). The Q-value function approximation is split into two streams by the duelling architecture, one for the state value and one for the advantage value. Without regard to the selected action, this enables the network to learn which states are valuable. By separating the action selection and Q-value estimate, double DQN tackles the overestimation problem in classic DQN. Dueling DDQN was able to outperform both Dueling DQN and Double DQN by combining these two strategies. By employing the dueling architecture to separate the value estimation of each action from the state estimation, it overcomes the overestimation problem of DQN and the errors in action values. Combining Dueling DQN and Double DQN results in Duelling DDQN. In this approach, the value state and benefit of each action are estimated using two different networks. The expected reward from a given state is known as the value state, whereas the advantage is the difference between the expected reward from a single action and the expected reward from all actions in that state. Separate streams are used by the two networks employed in Dueling DDQN to estimate value state and advantage. To estimate the Q-value for each action in a specific state, the output from each stream is then merged. To aid with convergence, the advantage values are centered around zero. Similar to Double DQN, Dueling DDQN's update equation also uses estimates from distinct value and advantage streams. The Bellman equation is used the algorithm to update the value and advantage streams, and after combining them, it estimates the Q-value for each action in a particular state. The ultimate objective of Dueling DDQN is to learn an optimal policy that maximizes the predicted cumulative reward over time, just like other learning algorithms. Similar to DQN and Double DQN, the method is trained through an iterative process in which the agent observes the current state, chooses an action based on the estimated Q-values, gets rewarded, and updates the value and advantage streams to enhance the estimates of the Q-values. Duelling DDQN is a

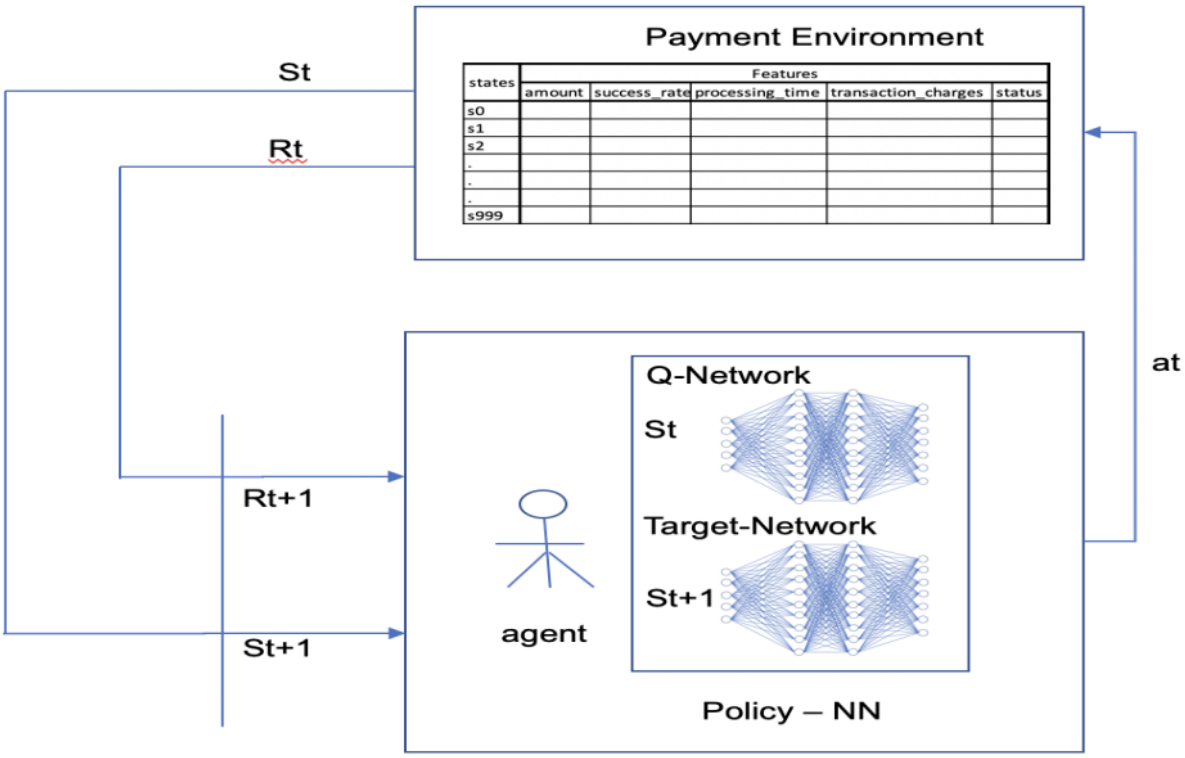


Figure 1: In DDQN, two separate networks are used to overcome the overestimation problem of Q-values in DQN. The first network is used for selecting the best action in the next state while the second network is used to evaluate the Q-value of that action. Both networks have the same architecture, and the target Q-values are calculated using the second network.

potential strategy for enhancing the performance of Q-learning algorithms in complicated environments since it has been demonstrated to outperform both DQN and Double DQN on a number of benchmark tests. Using prioritized experience replay, learning from numerous frames (rather than just one), or combining additional methods like distributional RL or Noisy Nets are some potential solutions to the Duelling DDQN's drawbacks.

7.5 Dueling DDQN with Thompson Sampling

A probabilistic algorithm for making decisions in the face of uncertainty is Thompson Sampling. It has been modified for use in reinforcement learning contexts and is frequently used in multi-armed bandit situations. By keeping a distribution over the optimal actions and choosing actions based on the likelihood that they will be optimal, Thompson Sampling resolves the exploration-exploitation conundrum. The problem of simultaneously optimizing several objectives can be addressed by combining Thompson Sampling and the Dueling Deep Q-Networks (DDQN) method in the context of multi-objective reinforcement learning (MORL). An outline of the procedures for employing Thompson Sampling with Dueling DDQN is provided below:

1. Specified the various objectives to optimize. We are maximizing the success rate and minimizing the transaction charges and processing time.
2. Using the Dueling DDQN architecture, which separates the estimate of the state-value function (V) from the action-value function (Q), to implement the Dueling DDQN architecture.

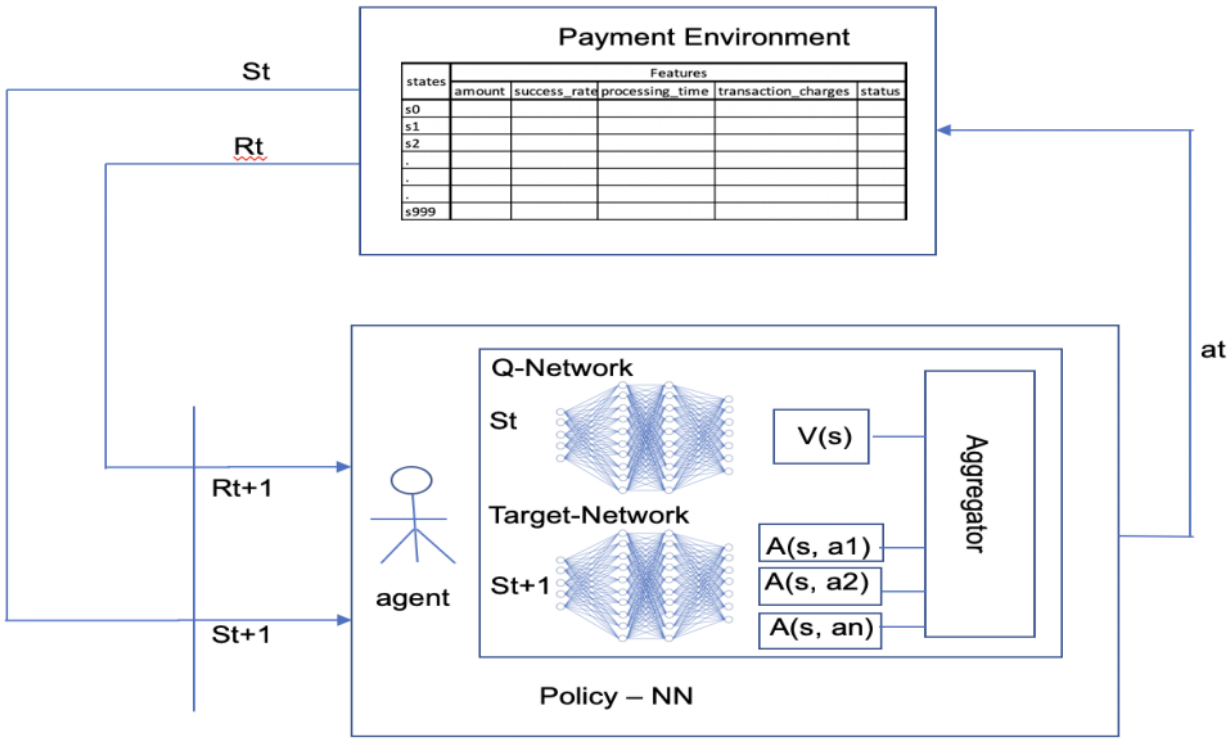


Figure 20: In duelling DDQN, the network is divided into two streams: one for estimating the state value function and another for estimat the advantage function. The two streams are then combined to obtain the Q-values for each action. This allows the network to separately learn the value of being in a particular state and the advantage of taking each action.

3. Initializing Thompson Sampling: Created a probability distribution for each objective. The belief in the actual action values for each objective is represented by these distributions. For finite objectives, the beta distribution is a popular option, while for continuous targets, the Gaussian distribution.
4. Exploration-Exploitation using Thompson Sampling: During acti election, selected a sample action from each objective's distribution using Thompson Sampling rather than the action with the highest estimated action value. The distributions and their uncertainty serve as the foundation for the sampling action probability.
5. Executing Selected Actions in the Environment and Watching Rewards: Carry out the Selected Actions in the Environment and monitored the Rewards Related to Each Objective.
6. Update Thompson Sampling Distributions: After obtaining the rewards, employed Bayesian inference to update the Thompson Sampling distributions for each objective. To improve the estimate of the genuine action values, incorporated the observed rewards into the distributions. This update procedure aids in modifying the distributions in light of recent information.
7. Dueling DDQN Update: Using the observed transitions (state, action, reward, and next state) and the appropriate target values, update the Dueling DDQN network's parameters. The Bellman equation and the estimated action-values derived from the Thompson Sampling
8. Iterative Process: Carried out steps 4 through 7 repeatedly for every new episode or iteration. The ideal actions and their associated values for each objective will be more accurately represented as the Dueling DDQN network and Thompson Sampling distributions converge over time.

While Dueling DDQN offers a potent architecture for estimating both the state-value and action-value functions, Thompson Sampling enables adaptive exploration by sampling actions based on the uncertainty of the estimated action values. In Multi-Objective-Reinforcement Learning (MORL) issues, this combination enables the agent to learn efficient action-selection rules that take into account the trade-off between several objectives.

8 Training of RL Agents

The analysis of the average reward or total reward can be used to assess the efficiency of learning, which is a fundamental component of reinforcement learning. These measurements offer perceptions into how well the learning algorithm performs and aid in evaluating its development and success in completing the given goal. We can better assess how well the algorithm is learning and whether it is making real progress toward reaching the required objectives by tracking and examining the rewards gained during training.

1. Tracking Avg Rewards Over episodes: Epsilon starts at a value of 1.0 and gradually decreases at a rate of 0.01 throughout training. This epsilon-greedy technique enables the agent to initially explore the surroundings before shifting to exploitation later on. The diagram effectively depicts this change, showing how after the 40th episode the agent begins to place a higher priority on exploitation than exploration, leading to notable performance gains.

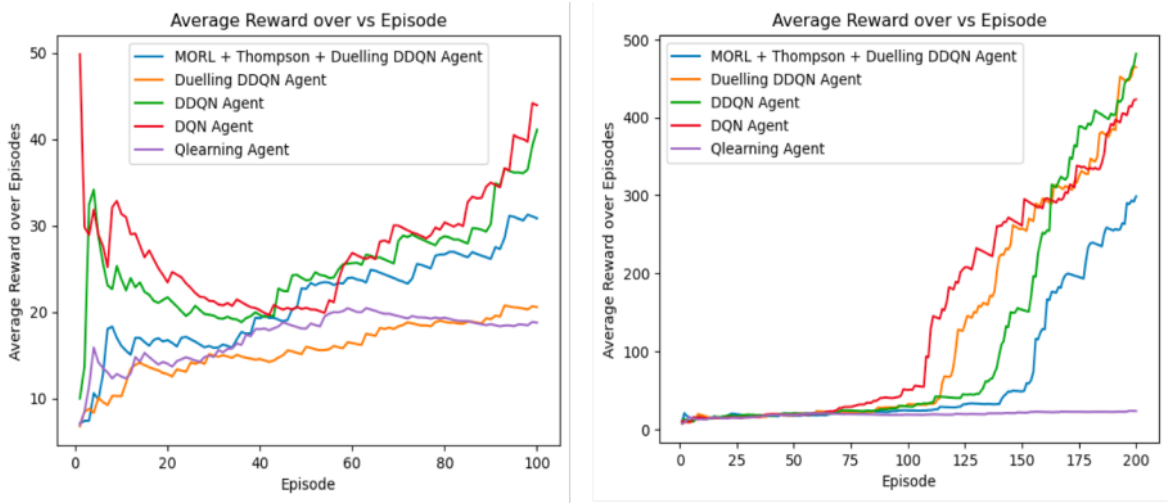


Figure 21: Average Rewards Over Episodes: The graph displays the typical incentives received by various agents across episodes with a total of 100 to 200. It offers insightful information on the agents' performance and development as learners over the course of this particular episode range.

2. Tracking Total Rewards Over episodes There is an observable shift from exploration to exploitation in the overall rewards plot for episodes 100 to 200. As the bot investigates various activities and picks up information from the environment, the overall rewards initially display variability and fluctuations. The overall rewards, however, show a clear upward trend as the episodes go on, suggesting the agent's growing capacity to amass bigger rewards through utilization of its acquired knowledge. This change denotes the agent's enhanced decision-making abilities and its capacity to maximize rewards by selecting actions based on better information. Overall, the rise in total incentives attests to the success of the training program and the agent's capacity for learning.

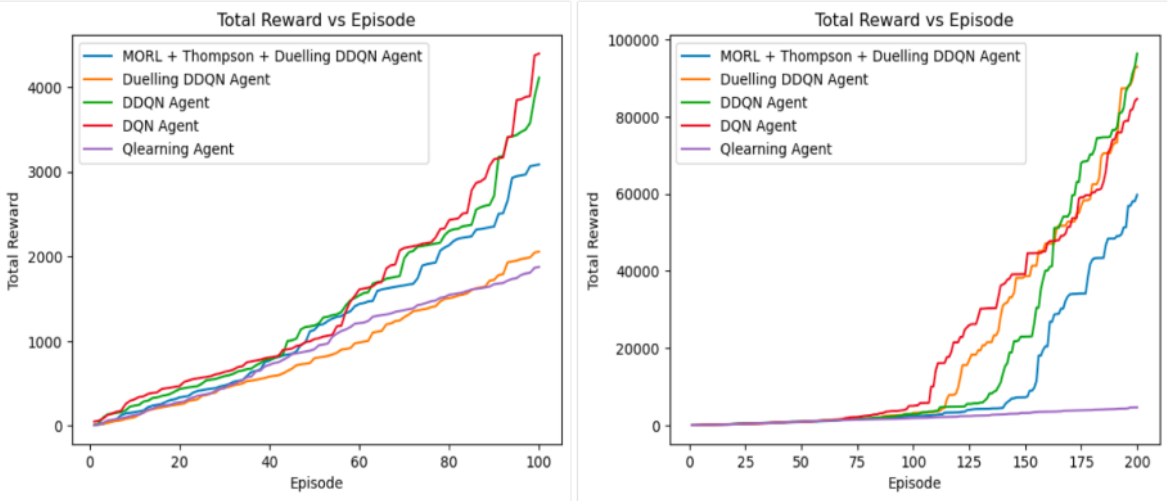


Figure 22: Total Rewards Across Episodes: The graph shows the total prizes the agents have earned throughout the course of the 100–200 episodes. In addition to giving an overall assessment of the agents' performance and efficiency on the assignment, it highlights the cumulative incentives the agents have received.

3. Tracking episodic Rewards Over episodes The diagram shows how episodic incentives grow over time, illuminating a noteworthy pattern noticed across all agents. The episodic incentives for each agent gradually increase after the 40th episode. This pattern shows that as the episodes go on, the agents are picking up new information and develop their decision-making skills. The agents' exploration phase is shown by the substantially smaller dispersion of episodic rewards in the early episodes. The agents actively experiment with various activities during this phase, learning from the results, which yields a range of incentives. However, as the episodes go on, the episodic rewards increasingly rise and become more reliable, signifying the agents' shift towards using the information they have learnt. After the 40th episode, episodic rewards began to rise, indicating that the agents had developed useful tactics and were making more profitable choices. It represents their growth as learners and their capacity to obtain greater rewards as they acquire more environmental experience. Overall, the graphic shows how the agents have a favourable learning curve and receive more episodic rewards as they progress through the episodes.

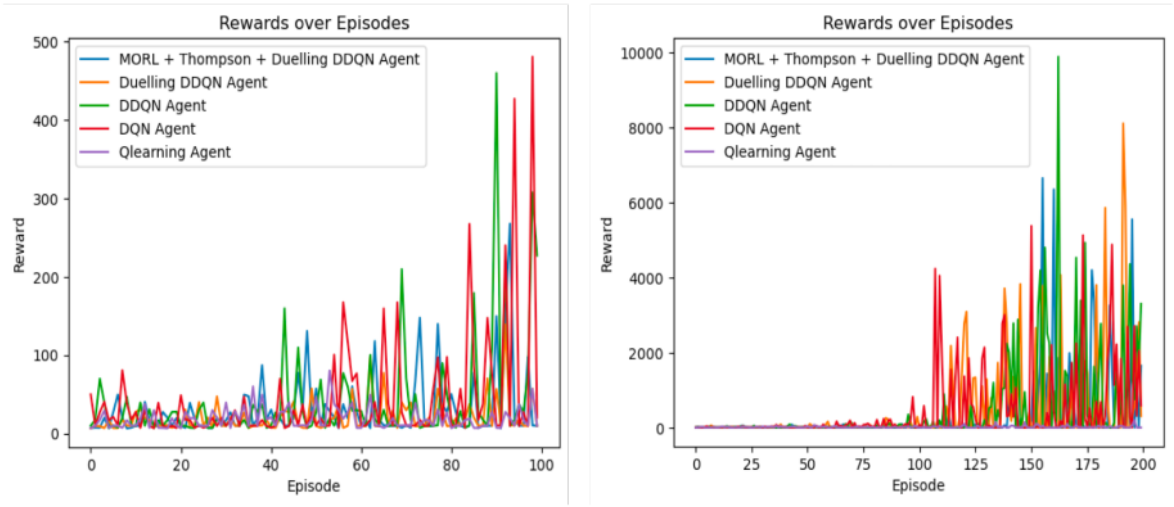


Figure 23: Rewards From Episodes Over Episodes: The figure shows the episodic prizes that the agents received over the course of 100 to 200 episodes. It provides a detailed look of the agents' performance and the variances in rewards throughout this episode range by showcasing the awards attained in each unique episode.

4. Training Summary: It is clear from the summary below that while the DQN agent originally had a larger average reward, over the course of episodes, the Duelling DDQN agent consistently outperformed all other agents in terms of average reward. This indicates that the Duelling Double DQN algorithm has done a better job of adjusting and enhancing its performance over time. The Duelling DDQN agent has learnt better methods and is making more optimal decisions in comparison to the other agents, as seen by the higher average payout. It emphasizes the benefit of applying the Duelling Double DQN strategy in this specific case.

Average Reward			
RL Agent	100 episodes	200 episodes	300 episodes
Thompson with Duelling DDQN	30.83	298.66	534.20
Duelling DDQN	20.55	464.53	775.64
DDQN	41.08	481.54	596.88
DQN	43.91	423.06	562.07
QLearning	18.75	23.20	25.82

Table 1: Comparison of the average Rewards received by various RL agents across episodes.

9 Results

1. Duelling DDQN Avg Reward: Dueling DQN is a modification of the standard DDQN algorithm created to address the issue of DDQN networks' propensity to overstate the Q-values of individual actions. Dueling DQN can more precisely calculate the value of each action by dividing the Q-value function into separate streams for the state-value function and the advantage function. We compared the average reward attained by several algorithms as part of our evaluation. After the 300th episode, we noticed that Dueling DQN fared better than the others since its average prize was higher than the others'. This suggests that Dueling DQN was more successful at increasing prizes and discovering the best policies for the payment problem.

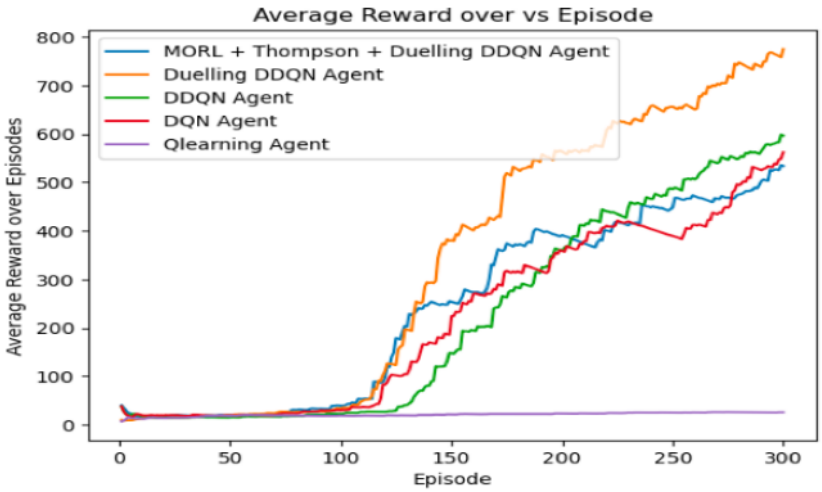


Figure 24: Dueling DQN has outperformed other algorithms, including DQN, DDQN, and Q-learning, at the 300th episode. The average reward achieved by Dueling DQN is significantly higher compared to the other algorithms, indicating its superior performance in maximizing rewards and learning optimal strategies for the payment problem.

2. Evaluation: The dataset is split into training and testing sets using an 80:20 ratio for evaluation. Utilizing the predict method from the trained model, the testing set is used to confirm the status of each payment type. This enables us to get the expected payment kinds based on the traits and patterns that the model has learned. To evaluate the model's effectiveness and accuracy in predicting the state of various payment kinds, the projected payment types can then be compared with the actual status in the testing set.
3. Comparision Summary: The below table compares the expected values with the testing data to assess the success rate, processing time, and transaction fees for each payment method. The table sheds light on how accurate the model is at predicting these parameters. It enables evaluation of the model's performance in raising the success rate for the optimal payment types while lowering transaction costs and processing times for each type of payment.

Overall summary of type, proc time and tran char						
Payment Type	No.Type	No.Predicted Type	Proc. Time	Predicted Proc. Time	Tran Chg	Predicted. Tran Chg
0	6.0	1.0	0.47	0.28	0.00	0.00
1	2.0	0.0	0.19	0.0	0.50	0.0
2	2.0	2.0	0.47	0.83	0.54	0.46
3	1.0	8.0	0.89	0.24	0.5	0.49
4	4.0	1.0	0.43	0.49	0.42	0.42
5	1.0	0.0	0.10	0.00	1.0	0.0
6	0.0	4.0	0.00	0.62	0.00	0.00

Table 2: Comparison of the existing versus predicted payment type, processing time and transaction charges

4. success rate: We can see some intriguing findings in the table above. In contrast to the learnt dueling DDQN agent's prediction of eight records, the testing data for payment type 3 only had one record. This implies that the agent is coming up with several recommendations for this kind of payment, possibly indicating a deeper degree of investigation. The number of records for payment type 6 has increased from zero in the testing data to four in the forecasts provided by the dueling DDQN agent, on the other hand. This suggests that based on the training data, the agent has developed the ability to produce suggestions for this payment type. The agent proposes leaving the advice for payment type 2 alone in terms of suggestions. The recommended for payment type 0 is decreased from 6 to 1, and for payment type 1, it is decreased from 2 to 0. These modifications show that the agent has mastered the ability to modify its advice in accordance with the particulars of each payment method. Overall, our findings show that the duelling DDQN agent can modify its recommendations according on the form of payment, perhaps producing more precise and specific advice for each situation.
5. processing time: For evaluation of processing time, compared the processing times of the current records in the test data with the anticipated data for each payment type in order to assess the processing time. The objective was to ascertain whether processing times for the projected payment kinds were faster than those for the original payment types. When processing times were compared, it was found that the test records' processing times totalled 2.57 whereas the projected payment types' processing times totalled 2.47. It has been noted that the anticipated payment methods reduced the total processing time by about 3.88% overall. This shows that the predictions made by the model were successful in identifying the payment kinds that would eventually result in faster processing times. A 3.73% reduction in processing time can make a big difference, especially in situations when
6. transaction charges: Similar to processing time, evaluated the transaction charges for the test records vs the records of the predicted payment type. Analysis revealed that the test records' total transaction charges were 2.97, but the records with the expected payment types accrued transaction charges of 1.39. These data can be compared to see that there was a huge reduction of 53.12% change in transaction fees.

10 Conclusion

In conclusion, the prediction accuracy and overall performance may be improved by further training the dueling DDQN algorithm over a larger number of episodes. It is clear that the duelling DDQN algorithm has had a substantial impact when comparing the success rate, processing time, and transaction fees. It has successfully increased each payment type's success percentage while concurrently reducing transaction fees and processing times. More episodes can be used to train the algorithm, which will produce predictions that are even better and will further improve the success rate, processing time, and transaction fees. Businesses may experience significant effects from this constant improvement since it can boost productivity, cut expenses, and raise customer happiness. The learning and adaptability of the duelling DDQN algorithm has shown its capability for managing various payment kinds efficiently. Real-world applications benefit from its ability to maximize success rates while lowering transaction costs and processing time. The creation of more sophisticated and effective payment processing systems can be facilitated by more study and advancement in reinforcement learning algorithms like duelling DDQN.

10.1 Future Scope

This paper includes a comparison of the five algorithms

1. Q-Networks
2. Deep Q-Networks
3. Double Deep Q-Networks
4. Duelling Double Deep Q-Networks
5. Thompson Sampling with Duelling Double Deep Q-Networks

The potential for considerable breakthroughs in payment environments employing Multi-Objective Time-Sensitive (MOTS) algorithms is bright. A major area of study is the creation of intelligent systems that can simultaneously increase payment success rates while reducing processing times and transaction costs. Using MOTS algorithms, which were created expressly for the payment environment, it is now possible to simultaneously optimize various goals.[3] There are many advantages to incorporating MOTS algorithms into the payment system. It facilitates enhanced resource allocation and decision-making, which ultimately improves customer satisfaction. When it comes to payment processing, these algorithms can help in making more educated and effective decisions, which will enhance success

rates and decrease transaction costs and processing times. It is critical to recognize that Reinforcement Learning (RL) has unique difficulties, such as sample inefficiency, instability, and scaling problems when working with expansive state and action spaces. Despite these difficulties, research on RL is still underway, and efforts are being made to remove these drawbacks and broaden its usefulness in a variety of fields, including payment systems. Overall, the use of MOTS algorithms and ongoing improvements in RL approaches are key to the future of payment settings. These developments have the ability to completely transform the payments sector by maximizing a variety of goals, streamlining procedures, and improving overall effectiveness and client happiness.

References

[1] Ramya Bygari, Aayush Gupta, Shashwat Raghuvanshi, Aakanksha Bapna, and Birendra Sahu. An ai-powered smart routing solution for payment systems. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2026–2033, 2021.

[2] Oren Levy, Ronen Morecki, and Ben Yaniv Chechik. Computerized transaction routing system and methods useful in conjunction therewith, August 3 2021. US Patent 11,080,711.

[3] Kristof Van Moffaert, Tim Brys, Arjun Chandra, Lukas Esterle, Peter R Lewis, and Ann Nowé. A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In *2014 International joint conference on neural networks (IJCNN)*, pages 2306–2314. IEEE, 2014.

[4] Jesús Téllez Isaac and Sherali Zeadally. Design, implementation, and performance analysis of a secure payment protocol in a payment gateway centric model. *Computing*, 96:587–611, 2014.

Vidyadhar

ORIGINALITY REPORT

12%	7%	8%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to South Dakota Board of Regents Student Paper	1%
2	Submitted to Cranfield University Student Paper	<1%
3	"Deep Reinforcement Learning", Springer Science and Business Media LLC, 2020 Publication	<1%
4	Submitted to University of Birmingham Student Paper	<1%
5	Nam H. Chu, Diep N. Nguyen, Dinh Thai Hoang, Quoc-Viet Pham, Khoa T. Phan, Won-Joo Hwang, Eryk Dutkiewicz. "AI-enabled mm-Waveform Configuration for Autonomous Vehicles with Integrated Communication and Sensing", IEEE Internet of Things Journal, 2023 Publication	<1%
6	Submitted to Georgia Institute of Technology Main Campus Student Paper	<1%
7	auai.org Internet Source	

		<1%
8	hdl.handle.net Internet Source	<1%
9	Submitted to King's College Student Paper	<1%
10	d-nb.info Internet Source	<1%
11	www.frontiersin.org Internet Source	<1%
12	www.statworx.com Internet Source	<1%
13	amslaurea.unibo.it Internet Source	<1%
14	dspace.mit.edu Internet Source	<1%
15	repository.charlotte.edu Internet Source	<1%
16	JOST SCHATZMANN, KARL WEILHAMMER, MATT STUTTLE, STEVE YOUNG. "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies", The Knowledge Engineering Review, 2006 Publication	<1%

17	Bishoy Sabry, Mariam Nabil, Ghada Alsuhli, Karim Banawan, Karim Seddik. "Self-Optimized Agent for Load Balancing and Energy Efficiency: A Reinforcement Learning Framework with Hybrid Action Space", Institute of Electrical and Electronics Engineers (IEEE), 2023 Publication	<1 %
18	Submitted to University of Strathclyde Student Paper	<1 %
19	aledan.ece.illinois.edu Internet Source	<1 %
20	www.politesi.polimi.it Internet Source	<1 %
21	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
22	cse.iitkgp.ac.in Internet Source	<1 %
23	epub.ub.uni-muenchen.de Internet Source	<1 %
24	"Artificial General Intelligence", Springer Science and Business Media LLC, 2023 Publication	<1 %
25	testlify.com Internet Source	<1 %

26	Submitted to University of Durham Student Paper	<1%
27	www.mdpi.com Internet Source	<1%
28	Edmund Durfee. "Computationally-efficient combinatorial auctions for resource allocation in weakly-coupled MDPs", Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS 05 AAMAS 05, 2005 Publication	<1%
29	Guillem Muñoz, Cristina Barrado, Ender Çetin, Esther Salami. "Deep Reinforcement Learning for Drone Delivery", Drones, 2019 Publication	<1%
30	www.tnt.uni-hannover.de Internet Source	<1%
31	Rieker, Jeffrey D., and John W. Labadie. "An intelligent agent for optimal river-reservoir system management", Water Resources Research, 2012. Publication	<1%
32	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1%
33	Submitted to The Robert Gordon University Student Paper	<1%

34	www.researchgate.net Internet Source	<1%
35	Submitted to National University of Ireland, Galway Student Paper	<1%
36	Submitted to Portland State University Student Paper	<1%
37	lume.ufrgs.br Internet Source	<1%
38	Submitted to AUT University Student Paper	<1%
39	Fang Wang, Renjun Feng, Haiyan Chen, Wen Wu, Fei Zhu. "An information network security policy learning algorithm based on Sarsa with optimistic initial values", International Journal of Computational Science and Engineering, 2019 Publication	<1%
40	Submitted to Leiden University Student Paper	<1%
41	Submitted to Middle East Technical University Student Paper	<1%
42	Uragami, Daisuke, Yu Kohno, and Tatsuji Takahashi. "Robotic Action Acquisition with Cognitive Biases in Coarse-grained State Space", Biosystems, 2016.	<1%

Publication

43	Submitted to Westminster International University in Tashkent Student Paper	<1 %
44	Xue Tao, Min Jiang, Yumeng Liu, Qi Hu et al. "Predicting three-month fasting blood glucose and glycated hemoglobin of patients with type 2 diabetes based on multiple machine learning algorithms", Research Square Platform LLC, 2022 Publication	<1 %
45	Yoshihisa Tsurumine, Yunduan Cui, Eiji Uchibe, Takamitsu Matsubara. "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation", Robotics and Autonomous Systems, 2018 Publication	<1 %
46	lup.lub.lu.se Internet Source	<1 %
47	www2.mdpi.com Internet Source	<1 %
48	Submitted to University of Leeds Student Paper	<1 %
49	Submitted to University of North Texas Student Paper	<1 %

50	Weiwei Zhao, Hairong Chu, Xikui Miao, Lihong Guo, Honghai Shen, Chenhao Zhu, Feng Zhang, Dongxin Liang. "Research on the Multiagent Joint Proximal Policy Optimization Algorithm Controlling Cooperative Fixed-Wing UAV Obstacle Avoidance", Sensors, 2020 Publication	<1 %
51	nrl.northumbria.ac.uk Internet Source	<1 %
52	Submitted to College of Engineering & Technology Bhubaneswar Student Paper	<1 %
53	Submitted to Gitam University Student Paper	<1 %
54	Hiren Kumar Thakkar, Ankit Desai, Priyanka Singh, Kamma Samhitha. "Chapter 30 ReLearner: A Reinforcement Learning-Based Self Driving Car Model Using Gym Environment", Springer Science and Business Media LLC, 2022 Publication	<1 %
55	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1 %
56	Submitted to Monash University Student Paper	<1 %

57	Submitted to National University of Singapore Student Paper	<1 %
58	Submitted to University of Bristol Student Paper	<1 %
59	D.H. Ballard. "Advances in reinforcement learning and their implications for intelligent control", Proceedings 5th IEEE International Symposium on Intelligent Control 1990, 1990 Publication	<1 %
60	Farhang Sahba. "Application of reinforcement learning for segmentation of transrectal ultrasound images", BMC Medical Imaging, 2008 Publication	<1 %
61	Submitted to International Islamic University Malaysia Student Paper	<1 %
62	Submitted to Kocaeli Üniversitesi Student Paper	<1 %
63	Submitted to Liverpool John Moores University Student Paper	<1 %
64	Submitted to University of Auckland Student Paper	<1 %
65	Submitted to University of London External System	<1 %

Student Paper

66	Submitted to University of Sheffield Student Paper	<1%
67	Submitted to University of South Florida Student Paper	<1%
68	Submitted to University of Ulster Student Paper	<1%
69	Xian-Da Zhang. "A Matrix Algebra Approach to Artificial Intelligence", Springer Science and Business Media LLC, 2020 Publication	<1%
70	scholars.unh.edu Internet Source	<1%
71	C. Clausen, H. Wechsler. "Quad-Q-learning", IEEE Transactions on Neural Networks, 2000 Publication	<1%
72	Fabian Fernando Jurado Lasso, Mohammadreza Barzegaran, Jesus Fabian Jurado, Xenofon Fafoutis. "ELISE: A Reinforcement Learning Framework to Optimize the Sloftframe Size of the TSCH Protocol in IoT Networks", Institute of Electrical and Electronics Engineers (IEEE), 2023 Publication	<1%
73	Submitted to University of Exeter Student Paper	

		<1%
74	biblio.vub.ac.be Internet Source	<1%
75	download.atlantis-press.com Internet Source	<1%
76	www.flipkart.com Internet Source	<1%
77	"Advances in Intelligent Data Analysis IX", Springer Science and Business Media LLC, 2010 Publication	<1%
78	Thomas Gabel, Martin Riedmiller. "Chapter 18 CBR for State Value Function Approximation in Reinforcement Learning", Springer Science and Business Media LLC, 2005 Publication	<1%
79	V Akila, J Anita Christaline, A Jothi Mani, K Meenakshi. "Reinforcement Learning For Walking Robot", IOP Conference Series: Materials Science and Engineering, 2021 Publication	<1%
80	liacs.leidenuniv.nl Internet Source	<1%
81	link.springer.com Internet Source	<1%

82	Chunxi Tan, Ruijian Han, Rougang Ye, Kani Chen. "Adaptive Learning Recommendation Strategy Based on Deep Q-learning", Applied Psychological Measurement, 2019 Publication	<1 %
83	Krishnan Padmanabhan. "Performance analysis of redundant-path networks for multiprocessor systems", ACM Transactions on Computer Systems, 5/1/1985 Publication	<1 %
84	Submitted to Roehampton University Student Paper	<1 %
85	Submitted to University of Pretoria Student Paper	<1 %
86	Yuhan Su, Minghui Liwang, Seyyedali Hosseinalipour, Lianfen Huang, Huaiyu Dai. "Cooperative Relaying and Power Control for UAV-Assisted Vehicular Networks with Deep Q-Network", 2021 IEEE/CIC International Conference on Communications in China (ICCC), 2021 Publication	<1 %
87	arxiv.org Internet Source	<1 %
88	conference.fdiba.tu-sofia.bg Internet Source	<1 %
	etheses.whiterose.ac.uk	

89	Internet Source	<1%
90	iranarze.ir Internet Source	<1%
91	tel.archives-ouvertes.fr Internet Source	<1%
92	www.arxiv-vanity.com Internet Source	<1%
93	www.cs.uu.nl Internet Source	<1%
94	"Clique-based cooperative multiagent reinforcement learning using factor graphs", IEEE/CAA Journal of Automatica Sinica, 2014 Publication	<1%
95	Submitted to Coventry University Student Paper	<1%
96	Engin Ipek. "Self-Optimizing Memory Controllers", ACM SIGARCH Computer Architecture News, 06/01/2008 Publication	<1%
97	F. Richard Yu, Ying He. "Deep Reinforcement Learning for Wireless Networks", Springer Science and Business Media LLC, 2019 Publication	<1%

98	Fengsheng Wei, Gang Feng, Yao Sun, Yatong Wang, Shuang Qin, Ying-Chang Liang. "Network Slice Reconfiguration by Exploiting Deep Reinforcement Learning with Large Action Space", IEEE Transactions on Network and Service Management, 2020 Publication	<1 %
99	International Series in Operations Research & Management Science, 2014. Publication	<1 %
100	Khoi Khac Nguyen, Trung Q. Duong, Ngo Anh Vien, Nhien-An Le-Khac, Minh-Nghia Nguyen. "Non-Cooperative Energy Efficient Power Allocation Game in D2D Communication: A Multi-Agent Deep Reinforcement Learning Approach", IEEE Access, 2019 Publication	<1 %
101	Lawal Babangida, Thinagaran Perumal, Norwati Mustapha, Razali Yaakob. "Internet of Things (IoT) Based Activity Recognition Strategies in Smart Homes: A Review", IEEE Sensors Journal, 2022 Publication	<1 %
102	Mark Denny. "Chapter 13. Biological Materials I Materials Mechanics", Walter de Gruyter GmbH, 2016 Publication	<1 %

103	Mohit K. Sharma, Alessio Zappone, Mohamad Assaad, Merouane Debbah, Spyridon Vassilaras. "Distributed Power Control for Large Energy Harvesting Networks: A Multi-Agent Deep Reinforcement Learning Approach", IEEE Transactions on Cognitive Communications and Networking, 2019 Publication	<1 %
104	Rasoul Amirzadeh, Asef Nazari, Dhananjay Thiruvady. "Applying Artificial Intelligence in Cryptocurrency Markets: A Survey", Algorithms, 2022 Publication	<1 %
105	Victor M. Saucedo, M.Nazmul Karim. "Real time optimal feeding in a fermentor using a Markov decision algorithm", Computers & Chemical Engineering, 1998 Publication	<1 %
106	Whitehead, S.D.. "Reinforcement learning of non-Markov decision processes", Artificial Intelligence, 199502 Publication	<1 %
107	Xiuqi Li, Jie Wu, Shi Zhong. "ISRL: intelligent search by reinforcement learning in unstructured peer-to-peer networks", International Journal of Parallel, Emergent and Distributed Systems, 2008 Publication	<1 %

108	assets.researchsquare.com Internet Source	<1 %
109	digital.ub.uni-paderborn.de Internet Source	<1 %
110	ebin.pub Internet Source	<1 %
111	epdf.pub Internet Source	<1 %
112	etheses.bham.ac.uk Internet Source	<1 %
113	incompleteideas.net Internet Source	<1 %
114	ipfs.io Internet Source	<1 %
115	jungfrau.tamu.edu Internet Source	<1 %
116	medium.com Internet Source	<1 %
117	public.hronopik.de Internet Source	<1 %
118	publications.polymtl.ca Internet Source	<1 %
119	repub.eur.nl Internet Source	<1 %

120	thinkmind.org Internet Source	<1%
121	www.dtic.mil Internet Source	<1%
122	www.informs-sim.org Internet Source	<1%
123	Razieh Mohammadi, Zahra Shirmohammadi. "DRDC: Deep reinforcement learning based duty cycle for energy harvesting body sensor node", Energy Reports, 2023 Publication	<1%
124	Doran Chakraborty. "Sample Efficient Multiagent Learning in the Presence of Markovian Agents", Springer Science and Business Media LLC, 2014 Publication	<1%
125	Sertan Girgin. "", IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics), 10/2007 Publication	<1%
126	livrepository.liverpool.ac.uk Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		