I.J. Modern Education and Computer Science, **2022, 3, 88-102**
Published Online June 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2022.03.06

Autonomous Taxi Driving Environment Using Reinforcement Learning Algorithms

Showkat A. Dar
Department of Computer Science and Engineering, Annamalai University, India
Email:showkatme2009@gmail.com

S. Palanivel
Department of Computer Science and Engineering, Annamalai University, India
Email: sapl_yughu@yahoo.com

M. Kalaiselvi Geetha
Department of Computer Science and Engineering, Annamalai University, India
Email: geesiv@gmail.com

Received: 04 January 2022; Accepted: 25 March 2022; Published: 08 June 2022

Abstract: Autonomous driving is predicted to alter the transportation industry in the near future. For decades, carmakers, researchers, and administrators have already been working in this sector, with tremendous development. Nevertheless, there are still many uncertainties and obstacles to solve, not only in terms of technical technology, as well as in terms of human consciousness, culture, and present traffic infrastructure. With respect to technological challenges, precise route identification, avoiding the improper location, time delay, erroneous drop-off, unsafe path, and automated navigation in the environment are only a few. RL (Reinforcement Learning) has evolved into a robust learning model which can learn about complications in high dimensional settings, owing to the advent of deep representation learning. Environment learning has been shown to reduce the required time delay, reduce cost of travel, and improve the performance of the agent by discovering a successful drop-off. The major goal is to ensure that an autonomous vehicle driving can reach passengers, pick them up, and transport them to drop-off points as quickly as possible. For performing this task, RL methods like DQNs (Deep Q Networks), Q-LNs (Q-Learning networks) , SARSAs (state action reward state actions), and ConvDQNs (convolution DQNs) are proposed for driving Taxis autonomously. RL agent's decisions are based on MDPs (Markov Decision Processes). The agent has effectively learnt the closest path, safety, and lower cost, gradually obtaining the capacity to travel bigger areas of the successful drop-off without negative incentive for reaching the target using these RL approaches. This scenario was chosen based on a set of requirements for simulating autonomous vehicles using RL algorithms. Results indicate that ConvDQNs are capable of successfully controlling cars in simulation environments than other RL methods. ConvDQNs are a combinations of CNNs (Convolution Neural Networks) and DQNs. These networks show better results than other methods as their combining of procedures gives improved results. Results indicate that ConvDQNs are capable of successfully controlling a car to navigate around a Taxi-v2 environment than the existing RL methods.

Index Terms: Environment learning, autonomous driving, reinforcement learning (RL), Q-learning, deep Q network (DQN), and state action reward state action (SARSA), convolution deep Q network (ConvDQN), and deep learning.

1. Introduction

Smart cities are emerging technologies of the next generation ICTs (information and communication technologies) that can help alleviate "big city diseases," co-ordinate urban developments, and improve operational efficiencies of city lives [1,2]. Intelligent transportations [3] are an essential component of smart cities aimed at enhancing transportation system's efficiency, maximize resources while ensuring traffic safety [4]. They are s crucial city lives and smooth operations as currently cities suffer from a range of issues including traffic congestions, accidents, energy wastes, polluted air where intelligent transportations could be a probable answer [5, 6].

Developments in the area of IoTs (Internet of Things) and wireless communication technology have been helpful in collecting trajectory records of mobile objects making intelligent transportations a possibility [7] where smart phones [8], private cars [9], and public transportations [10] are examples of devices that can use GPSs (Global Positioning Systems). Location data generate large amounts of trajectory data on a day to day basis and analyzing these voluminous digital traces can result in understanding city’s social and economic characteristics [11]. However, as the operational taxi counts increase, laying corresponding operational strategies face flaws and issues like finding them during peak hours, uneven distributions, and drivers’ denial of services [12]. The driver’s passenger-seeking strategies are empirical and differ significantly from one another [13, 14], resulting in low service efficiencies and remunerations.

Companies owning fleets of taxis used greedy geographical methods for completing orders like finding nearest driver-passenger pairs or employing first-come-first-serve queuing strategies [15]. Certain goals were visualized for improving efficiency of taxi services like reducing wait times and distances between drivers and customers for increasing profits and tracing potential customers [16]. Many studies on demands and supplies have examined optimal strategies including taxi recommender systems where average wait times, distances, fares, and transition probabilities were considered for recommending high revenue potential locations [17]. Overall result summaries indicated distances to next cruising locations and the wait times were significant factors for improving efficiencies. Luo et al [18] examined spatiotemporal characteristics of taxi route recommendations in terms of dynamic taxi route recommendations considering it as sequential decision-making issues and devised a two-step solution. Taxi monitors detected abnormal driving behaviours for improving taxi services [19]. Mobility spatiotemporal framework based on DMTs (data mining techniques) were proposed from taxi trajectories [20] where travel demands in different parts of cities were predicted in timely and efficient manners. These studies helped in gaining better understanding of factors that influence taxi services.

The technical challenges can be summed up as issues of accurate route detections, shortest distances, avoiding incorrect locations, time delays, incorrect drop-off points, risky routes, and automatic navigations. As a result of DLTs (deep learning techniques) and systems architectures using them for ATDs (Automatic Taxi Drives) have been proposed for high precisions. More than perceptions, most tasks of ATDs have lost their appropriateness in traditional supervised learning methods. ICTs [21, 22, 23] allow people to communicate more effectively and recent developmental researches in RLs have produced remarkable results in answering several technical challenges in ATDs including accurate route detections, avoiding wrong areas, reducing time delays, reducing incorrect drop-offs and risky routes. Hence, the main goal of this project is to ensure that ATDs reach passengers as quickly as possible reducing pick and transportation times. This work uses RL methods like Q-LNs, DQNs, SARSAs , and ConvDQNs for ATDs . These RL methods, the agent has successfully learned the nearest route with successful drop-off, shortest distance, wrong locations, safety and ability to navigate negative reward for reaching the destination after every Time step. The design is cost effective when run in real scenarios as shown in its simulated testing environments on Gym's Cab open AIs(Artificial Intelligences).

2. Background

Gao et al. [24] used RLs to improve taxi driving strategies and maximise global profits where taxi driving sequences were optimized using MDPs. Taxi locations and operation statuses were defined as states in this model and actions included operations like empty drives, carrying passengers, or waiting along with driving behaviours. Effective driving ratios were measured as cab driver's total profits in single working days and defined as rewards of objective functions evaluating driving policies. Q-LNs with maximum cumulative rewards learnt best choice for cab drivers in locations. The study’s experiments in Beijing using historical trajectory data showed enhancements in cab drivers' profits and efficiency while also making it easier for passengers to find taxis. This new model was advantages over previous methods as it was optimal and not based on prior knowledge.

Shou et al [25] used MDPs to model optimal sequential decisions in passengers engaging drivers using e-hailing drivers who just had to execute these orders. These drivers differed from traditional drivers and matched customers to corresponding platforms similar to TNCs (Transportation Network Companies). The proposed scheme’s simulated experiments on Monte Carlo were compared for optimality with drivers following baseline heuristics like hotspot strategies and thus the model's effectiveness was validated. Since, the count drivers were sufficient, unmatched order counts became negligible in their proposed MDP model which captured supply-demand ratios. The study’s dynamic adjustments amongst drivers order allowed competition calibrations of matches.

To answer taxi dispatch problems, Mao et al [26] explored unique model free DRL DRL (Deep Reinforcement Learning) architecture. When trip demands and taxi supplies were geographically or temporally uneven in transportation networks, their framework redistributed automobiles as rebalancing strategies. Their policies provide optimal dispatch strategies while their value functions estimated expected costs with time stamps using FFNNs (feed-forward neural networks). In terms of user priorities their results showed that their learnt policies produced superior balancing in levels of services or cancellations.

Wang et al. [27] proposed MDPs and learning based on DQNs with action searches to improve dispatches of ride-sharing platform drivers. The study used data acquired from DiDi ride-sharing platform train trips their assessments of dispatching agents. Large-scale dispatching systems often service multiple geographical regions with varying demand-supply conditions. CFPTs (Correlated Feature Progressive Transfers) were developed together with two existing approaches to boost learning flexibility and efficiency, allowing information transfer in both spatial and temporal realms. Moreover, the study’s policies learnt by transferring information from source- target cities or across temporal spaces within same cities outperforming policies learnt without transfer learning.

In large scale on-demand ride-hailing services, Xu et al [28] introduced revolutionary order dispatches to provide efficient optimizations of resource consumptions and user experiences from global and farsighted perspectives. Their investigations allocated orders to drivers in coordinated manners using centralised algorithms. The study handled problems using learning and planning approaches: 1) summarizing demands and supplies using spatiotemporal quantization and assessing driver's values in specific state; 2) real-time plans for driver-ordered-pairs were valued in terms of immediate rewards and future gains while dispatched were solved using combinatorial optimizations. The study's proposed technique was adopted in Didi Chuxing's production systems after their lengthy offline trials and online AB testing showed it significantly improved platform's efficiency.

Chen et al [29] established a system to handle demanding urban autonomous driving situations using DRLs. Their RL improved automatically where low-dimensional latent states were captured using specialised input representations and visual encoded. Their framework included many model-free deep DRLs along with boosting techniques for improved performances. In high-definition driving simulations the scheme was evaluated with dense surrounding cars. Their results demonstrated that their approach was capable of completing task better then defined baselines.

Verma et al [30] used RL which needed well-defined state and action spaces to learn from taxi trajectory data for generating status and action slots. They testing with actual drivers would be ideal and im practical to consider many drivers due to costs. Without any prior knowledge of the surroundings or taxi demand situations, RL agents can generate incomes equivalent to top 10% of drivers and in the case of many learning agents, the dynamics would be dramatically different.

Kiran et al. [31] summarised DRL techniques and provided taxonomies for ATD tasks and thus addressed important computing problems in the deployment of autonomous driving agents. It also distinguished neighbouring topics that were similar but not standard RL methods including behaviour clones, imitation learning, and inverse RLs . This paper discussed functions of simulators in training agents, as well as approaches for validating, testing current robust RL systems.

Rasheed et al. [32] proposed a NDRL (New Adversarial Deep Reinforcement Learning) method for improving the robustness of ATD dynamics when adversaries try to inject erroneous data into sensor readings of autonomous vehicles in order to disrupt autonomous car's on road safety and space optimality. The autonomous cars kept themselves guarded by maintaining safe distances. LSTMs-GANs (Generative Adversarial Networks) models computed expected distance variations as a result of car's actions which were fed as inputs to NDRL algorithm. The algorithm then tried minimize variations in distances

Jin et al. [33] reconstructed drivers' actions using MDPs which considered impacts after long-term actions. The state-action value function computed based on RL frameworks, which included dynamic programming and data expansions. Drivers could use value functions to determine best options and then quantify expected future rewards in given states. Using historical orders data from Chengdu, the study investigated spatial distribution of function values and illustrated how the models could improve driving rules. Subsequently on-demand platforms were simulated where their results showed that their new model outperformed existing benchmark approaches in terms of total incomes, response rates, and reduction of wait times.

Unlike prior research, which mostly focused on model-based approaches, this study looked at adopting policy-based DRL modelling free methods. The policy-based DRL model has superior performances, according to numerical studies.

3. Proposed Methodology

RL algorithm is used to solve several ATDs challenges and for operating simulated automobiles, policy-based RL approaches such as Q-LNs , DQNs, SARSAs, and ConvDQNs are used in this study. Text-based simulation is used to show the Taxi-v2 environment. The suggested technique takes into account extended sequences of driving activities in global settings to ensure that drivers get appropriate policies and achieve effective drop-offs. Long-term strategies meet the needs of present environments, resulting in better matches between drivers and passengers through shorter distances, successful drop-offs, and rectification of destinations. Simulations of services on-demand were generated for model's effectiveness and improving matched driver-passenger pairs. Many vehicles staged in different locations were needed for successful drop-offs, minimized wait times, and quickest routes during peak traffic hours to destinations in cities irrespective of the city and vehicle counts. ATDs using RL was developed to balance supplies and demands of taxis in various locations and improving taxi utilisation rates and meeting additional needs such as successful drop-offs, reduced timestamps, right destinations, and reduced customer wait times.

3.1. Simulation model

The taxi (rectangle) begins in an unknown place, then picks up a passenger at B, at which time it changes colour from yellow to green. The cab then arrives at drop-off site G, bringing the episode to a close. A single episode consists of a series of states, actions, and rewards that culminate in a terminal state. When the taxi driver has successfully picked up a customer and delivered them off at the right destination, the episode is over. The taxi aims to go through each episode in the fewest number of steps possible by crossing safe pathways (":") and avoiding barriers ("|"). Because the agent is driven by incentives, we must carefully consider the rewards and penalties, as well as their magnitudes. The environment diagram including rectangular box marked yellow as Taxi, pick and drop off among B,G , R, Y indicated as blue and pick is shown below in Fig.1.

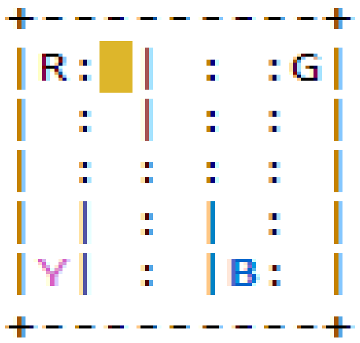

Fig.1. Environment diagram including Taxi, Pick and drop-off among (B, G, Y, R).

The following are some of the most critical considerations: I considered a few crucial factors:

- For a successful drop-off, the agent should earn a high positive reward;
- For the shortest distance travelled, the agent should receive a positive reward;
- If the agent tries to drop off a passenger in the wrong area, the agent should be punished.
- After each time step, the agent should receive a little negative incentive for failing to reach the destination..

The MDP model and RL approaches are used to complete the tasks listed above..

3.2. MDPs

MDPs are frameworks for sequential decision-making issues use RLs and parameters: S stands for state spaces that are infinite, A represents actionable states, P stands for probability matrices in the state's transitions, R represents rewards while γ represents discount factors in the interval [0,1]. If $s \in S$ represents an individual state for specific actions $a \in A$ in time t, then transition probability matrix for s during t + 1 i.e. $P(s, a, s')$ can be obtained using Equation (1)

$$P(s, a, s') \equiv \Pr(s_{t+1} = s'|s_t = s, a_t = a) \tag{1}$$

Where s' implies subsequent state while a implies actions. Moreover, MDPs select actions purely based on current states depicted in Equation (2),

$$\Pr(s_{t+1} = s'|s_0, a_0, s_1, a_1, \dots, s_t, a_t) = \Pr(s_{t+1} = s'|s_t = s, a_t = a) \tag{2}$$

MDPs use two functions namely V(s) for valuing states and Q(s,a) for valuing state-actions. The former represents probable cumulative future rewards of current states while the latter represents expected cumulative future rewards of state's actions. Traditional MDP agents adjust to their environments using policies π, which are maps from $\pi(s,a)$ to probabilities $\pi(a\,|\,s)$ of actions (a) taken in state (s). Hence, policies in MDP's value functions can be denoted as $V_\pi(s)$, $Q_\pi(s,a)$ and as depicted in Equations (3) and (4).

$$V_\pi(s) = E_x[(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots . + \gamma^{K-t-2} R_{K-1} + \gamma^{K-t-1} R_K)|s_t = s] = \mathbb{E}_\pi[\textstyle\sum_{k=t+1}^{K} \gamma^{k-t-1} R_k|s_t = s] \tag{3}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[\textstyle\sum_{k=t+1}^{K} \gamma^{k-t-1} R_k|s_t = s, a_t = a] \tag{4}$$

K is denoted as the time of the end state.Besides, the relation of $V_\pi(s)$ and $Q_\pi(s)$ are described as Equation (5),

$$V_\pi(s) = \textstyle\sum_{A}^{a\in A} \pi(a|s)\, Q_\pi(s, a) \tag{5}$$

Optimum strategy for increasing rewards is to simply maximize $V_\pi(s)$. The greedy method, as per Equation (5), is to choose the greatest of the $Q_\pi(s, a)$ (Equation (6)),

$$V_{\pi}(s) = \max_{a \in A} Q_{\pi}(s, a) \tag{6}$$

The actions of taxi drivers are determined by the current scenario rather than past ones [34]. They are primarily concerned with the current condition along with impact of present activity. Meanwhile, drivers are presumed to resourceful and sensible [35], then each decision is depending on computation and experience, implying every option is optimal strategy at the time. Drivers expressly admit instructions and transport people from one location to another when they believe the trip is warranted. After executing orders, drivers receive money by doing the activity of servicing the trip. Besides taking orders, drivers have the option of idling when there are no available or worthy orders. The drivers come to a halt or wander around in quest of vital orders. Drivers operate in accordance with their driving policies. As a result, consider of the drivers as MDP agents who operate in the environment according to a policy that describes how the agents choose behaviours to enhance reward. Modify the driving policies to enhance the long-term earnings of the drivers. Following that, the essential features of MDP driving behaviours are presented.

State Space: The driver's state is represented by a two-dimensional vector that represents spatiotemporal status. Every state is made up of two elements: time and place. Let $s = (t, l) \in S$ formally. As a result, $|S| = |T| \times |L|$. Considering discretizing the time-space into 10 minute increments and discretizing the location space by partitioning region as 500 m×500 m grid cells [36]. T is denoted as Time Space, L is denoted as location Space.l is denoted as the location of current state, l' is denoted as the location of next state, t is current state's time index, t' is next state's time index.

Action Space: In every condition, a driver can choose between two options: idle or driving. portraying the driver who is wandering or waiting in empty taxi location is the first action. They are not temporarily paired with any commands, allowing him/her to walk around freely. allocating a specific order to driver is the last action. Whenever a driver takes action serving, he or she cannot reverse his or her action till the order is completed. Set a = 0 and 1 in formal terms, with 0 denoting action idle and 1 denoting action driving.

Reward: Upon performing action and completing transition among states, a driver receives reward. The reward may be thought of as a cost for an order. For controlling degree to which MDP looks into future, a discount factor was developed. The greater element has a minor influence on the future. Because extended horizons provide a lot of volatility in the value function, it's best to employ a modest discount factor. Equation (7) is utilized to describe this function in a formal way.

$$R_{\gamma} = \sum_{t=0}^{Tr-1} \gamma^{t} \frac{r}{Tr} \tag{7}$$

R_{γ} is denoted as the reward when considering γ. r is denoted as the quoted price. T_r gives total transition steps consumed on an order trip.

State-action value function: since the optimize object, assesses driving policies by aggregating each action's rewards at every iteration. Actions have influences on not just the present benefits but also future rewards. It's crucial to think about the long-term payoff for global optimization. Equation (8) is the mathematical term for it,

$$Q(s, a) = E\left[\sum_{k=t}^{K} \gamma^{k-t} R_{\gamma}(s_k, a_k) | s_t = s, a_t = a\right] \tag{8}$$

reward of performing action a at state s is denoted as $R_{\gamma}(s, a)$. relation between $R_{\gamma}(s, a)$ and R_{γ} are described as Equation (9),

$$R_{\gamma}(s, a) = \begin{cases} R_{\gamma} & a = 1 \\ 0 & a = 0 \end{cases} \tag{9}$$

Policies: Distributions in states (s) to actions (a) in actionable spaces are stored in $\pi(a|s)$ where greedy policies trace optimality: $\pi(s) = \operatorname{argmax}_a Q(s, a)$

State value functions: Predicted cumulative rewards are received by drivers starting at states (s) following policies (π) till episodes end and their state function changed by $\pi(a|s)$ while learning leads to $V(s) = \max_{a \in A} Q(s, a)$

3.3. RL paradigm

An autonomous agent communicates with it's own surroundings in RL paradigm to learn how to improve its performance at a certain activity. A reward function R analyses an agent's performance rather than an expert telling them what to do. The agent chooses an action for each circumstance it encounters, and based on the usefulness of its decision, it may get a reward from its surroundings. Agent's aim is to raise overall amount of rewards it obtains over duration of its existence. major role of the agent in this work is to find a successful drop-off, shortest distance covered, correct locations, and lesser timestamp. By using what it has learned about predicted utility of alternative state actions agents increment their long term rewards. One major complexity in using RL is harmonizing explorations and exploitations where agents need to use their knowledge for selecting behaviours resulting in higher rewards. Agents need to risk unique actions that might fetch them higher rewards than existing best valued actions defined in the systems i.e. learning agents must use existing actions to get rewarded while research on unknown orders and take appropriate actions that can reward them higher.

Exploration entails gathering additional information about the surroundings, even if it necessitates taking a riskier activity. Exploitation refers to taking advantage of knowledge already gathered and choosing the most profitable route /path of action for successful drop-off. In an RL situation, the trade-off between explore and exploit must be properly adjusted. ϵ -greedy and softmax are two techniques that have been developed to address this trade-off. While using the widely used ϵ -greedy approach, an agent will choose an action randomly with probability $0 < \varepsilon < 1$ or chooses maximum valued action given present state greedily with probability 1- ε. Inherently, in start of the training phase, when less is known about issue environment, the agent must explore more. As the agent's training advances, he or she may begin to do more exploitation than exploration. Designing exploration tactics for RL agents is a major topic in research right now. There are two options for a taxi driver who has just dropped off a customer and is seeking for new business. They have the option of staying in the place and waiting for passengers, or they can go to a new place. Traveling to a new site incurs a cost, however if the new location is picked appropriately, it will have an influence on future journeys and the day's revenue. The following optimization approach is easily quantified using the RL model [36]. Policy learning is usually divided into two stages: policy evaluation and improvement. The optimization might be performed using a model-free method which is based on the MDP stated issue above. Policy evaluation denotes the agents' ongoing examination of the sample's value function, and policy improvement denotes the development of a better policy for maximizing rewards.

3.3.1. Q-LNs Network (QLN)

The Q-LNs Network (QLN) is a RLs (RL) technique for determining a given state's action value [37]. It doesn't need an environment model, and can tackle issues like stochastic transitions and incentives without the need for adaptations. Provided a current state, QLNs are RL policies that discover next optimal actions. The actions are selected randomly with the aim of increasing rewards where RL can discover optimal actions and making agents do those actions subsequently based on their environments.

Terms in Q-LNs

Some important terms of QLN are explained as,

States: State, s, indicates agent's current positions in environments.

Actions: Actions (a) are steps taken by agents in specific states (s).

Rewards: Agents gets rewarded for actions which can be positive or negative.

Episodes: the termination state of agents after which agents do not act.

Q-Values: The quality of actions performed specific states, s are obtained using Q-values i.e. Q: (s, a).

Temporal Differences: formulae used to find Q-Values with current and previous states and actions. Figure 1 shows reward achieved by the QLN process. The rewards are measured by varying the number of episodes from 0 to 1000 with 200 intervals. The average reward achieved by QLN is 10.0.

Policy evaluation: Taking specific policies, every agent assesses the relevant state value function and state-action value function as in policies evaluation phase. When taking action driving or action idle, the value function is modified. For ADTs , these two sorts of actions are carried out. The term "driving" refers to when a driver receives an order and proceeds to specified location. states' transition from state s to s' was completed by drivers, encompassing the geographical and temporal dimensions.

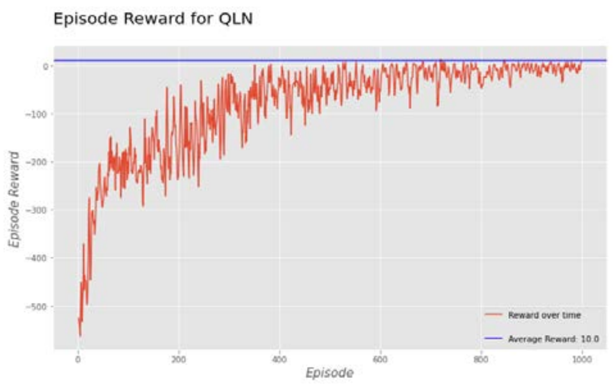

Fig. 2. Reward by QLN with respect to episode.

The Temporal-Difference (TD) update rule is given by Equation (10),

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R_\gamma + \gamma^{\Delta t} V(s') - Q(s,a)\right] \quad (10)$$

Present driver status is s = (t, l), and after finishing order state is s'=(t', l'). Temporal variation is represented by Δt. Δt is the total projected time for the pickup, waiting, and delivery processes, that is roughly equal to t'- t. When it comes to action idleness, an agent is permitted to go to adjacent places but will not be compensated right away. They can only travel across nine neighbour grids at a time. As a result, s' equals s + 1, and l' is now one of the l's neighbouring grids. Equation (11) is used to calculate the TD update rule

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\gamma V(s') - Q(s,a)] \quad (11)$$

α is denoted as learning rate.

Policy improvement: The optimization step of policy improvement is when the agents select the action with highest cumulative reward in the present state. To decide the appropriate action, the greedy technique was commonly adopted. $V(s) = \max_{a \in A} Q(s,a)$ for any spatiotemporal state s, in which Q (s, a) is learnt through policy assessment.

3.3.2. DQNs

DQNs are strategies for addressing situations with high-dimensional perceptual input that combine deep learning with Q-learning [38]. It's a multi-layered neural network that generates a projected future Q (s, a | θ) for every potential action, in which the network parameters are θ. To put it another way, To estimate the action value, DQN uses a neural network. The most recent four frames of data are fed to the first layer of DQNs for determining current states which are transformed into vectors of action values in connected layers [39]. Rewards of DQNs are shown in Figure 3. The rewards are measured by varying the number of episodes from 0 to 1000 with 200 intervals. The average reward achieved by DQN is 13.0.

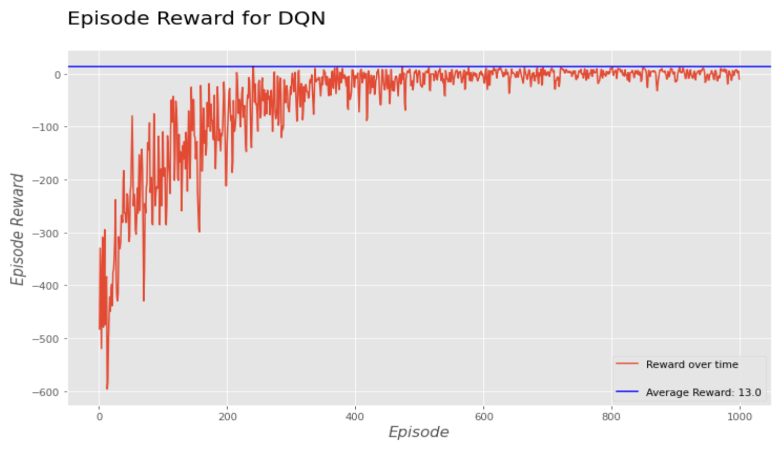

Fig. 3. Reward by DQN with respect to episode.

DQN minimizes a differentiable loss function L(θ) [39] by adjusting the network weights θ to improve the action value function.

$$L(\theta) = \left(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\theta) - Q(s_t, a_t|\theta)\right)^2 \quad (12)$$

On establishment of dynamic environments, current states of the partially observable environments can be expressed by models for their learning of policies.

3.3.3. SARSAs

SARSAs are MLTs (machine learning techniques) based on RL and learn policies generated by MDPs reinforcing majority functional updates of Q-values based on agent's current states (S1), actions (A1), rewards (R) received by the agent for agreeing to actions (A2) in state (S2) as its new state. SARSAs [40] are abbreviation for quintuple $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ and depicted as Equation (13),

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[R_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \quad (13)$$

Agents in SARSAs interact with their environments and update their policies based their executed actions (on-policy learning). Fig 4 shows the reward achieved by SARSAs processes. The rewards are measured by varying the number of episodes from 0 to 1000 with 200 intervals. The average reward achieved by SARSAs are -15.0.

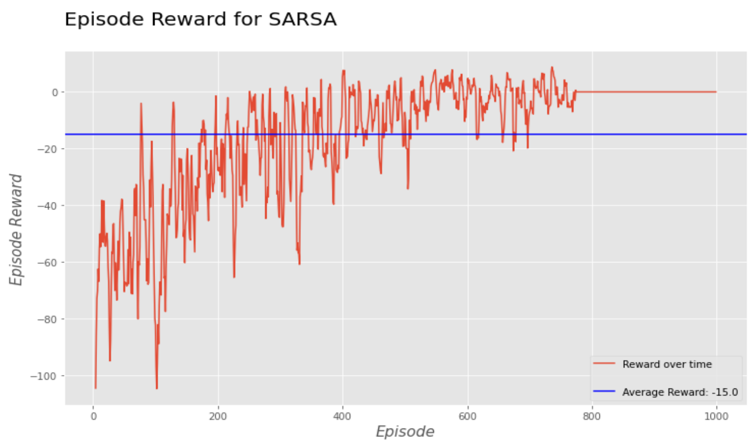

Fig. 4. Reward by DQN with respect to episode.

Q-values for state’s actions are updated by error adjustments of learning rates alpha. Q-values signify probable rewards received in next time steps for taking actions in states in addition to discounted future rewards to be received for subsequent state actions. Q-learning updates estimates of optimal state-action values function $\{Q^{*}\}$ based on maximum rewards for available actions. While SARSAs learns Q-values associated with policies, Watkin's Q-learning learn corresponding Q-values associated with optimal policies in explorations/exploitations of policies.

3.3.4. ConvDQNs

DNNs (Deep neural networks) are created based on fully connected CNNs to approximate the Q-function in terms of successful drop-off, time delays, distances, and costs. Three convolution layers and four dense layers make up the network. For each data type, zero padding 2D layers, convolution 2D layers, max pooling 2D layers, and fully-connected layers combine input types. The sane architecture is followed in targeted networks and values updated for every 1000 steps. NNs (neural Networds) are programmed to accept six actions with a set of rewards. The network is trained on a 32-piece mini-batch with a learning rate of 0.001. In the definition, the parameter description for approximating the Q function is discussed. When it comes to RLs, one has sparse and time-delayed labels for the rewards with successful drop-off, lesser time delay, lesser distance, and reduced cost. Only with these rewards the agent needs to figure out how to carry on in the environment. The architecture of the Network is as follows representing the activations and filters used at each hidden layer. The network outputs for every possible action’s Q-values. Fig. 5 shows the architecture of proposed ConvDQNs system.

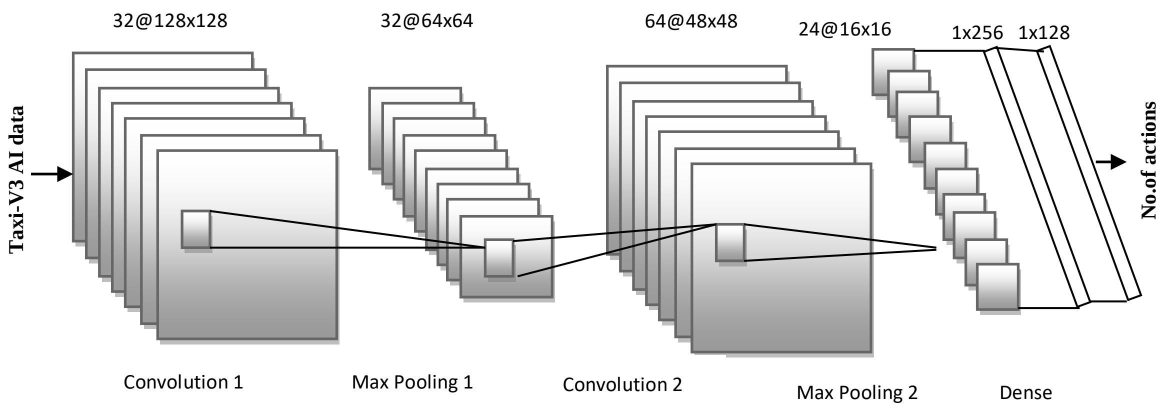

Fig. 5. Convolutional 2D connected network with DQN agent (CONVDQN).

This work,’s CNNs hidden layers use multiple 2D filtered output parameters namely Zero padded, convolutions, max pooled, and fully-connected. CNNs generate these layers with large number of parameters. The quality of the suggested model for ATDs for successful drop-offs will be determined by counts of layers and hidden parameters. In zero padding zeros are inserted in the beginning and end of taxi driving matrices and subsequent application of filters to inputs allows zero padded layers to have same output dimensions as described by equation (14),

$$\mathrm{zp} = \frac{(\mathrm{k}-1)}{2} \tag{14}$$

The filter size is denoted by the letter k. Convolution into 2D matrices is then performed via a convolution 2D layer with a sliding window 3*3 kernel size. With this 3*3 size, start with the little 3*3 matrices and work your way up to the larger matrices. Each neuron retrains using inputs containing the preceding layer's weights and biases. The max pooling 2D layers are generally placed after the convolution layers. Many characteristics, such as loop size and stride, are included in max pool layers. Two strides are used to build the max pooling 2D layers. The purpose of using max-pooling layers is to eliminate all maximum values in that filter and save time in the subsequent layer.To figure out how big a convolution layer's output should be by the equation (15),

$$\mathrm{os} = \frac{(\mathrm{w}-\mathrm{k}+2\mathrm{p})}{\mathrm{s}} + 1 \tag{15}$$

where os stands for w stands for output size, k stands for filter size, p stands for padding, and s stands for stride size. The softmax layer may then utilise this output to synthesise the data as desired. The features from the convolution and pooling layers will be classified in this layer. Dropout layer is a regularisation approach that tries to improve our model's prediction performance while also preventing overfitting. The model In the dropout layer, neurons in a layer will be deactivated at random with probability p. If the dropout value of a layer is changed and the training period is increased, the NNs learn about different, redundant representations quicker. To test the model, the dropout values in this study are integers ranging from 0 to 1.

Furthermore, during the implementation of CNN, ReLUs (rectified linear units) play an important role as activation functions. The activation function of the ReLU is given by the equation (16),

$$\mathrm{f(x)} = \max(0, \mathrm{x}) \tag{16}$$

wherein x is total inputs of NNs. The last layer of CNN design is the softmax (normalised exponential function) layer, which is a fully linked layer that consists of a logistic function expressed by an equation (17),

$$\sigma(\mathrm{z})_{\mathrm{i}} = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}, \mathrm{j} = 1, \ldots \mathrm{K} \tag{17}$$

where z is the input vector with K-dimensional vector, K-dimensional vector σ(z) is real values in the range (0, 1) and jth class is the predicted probability from sample vector x. After the completion of the output layer, squared loss can be used for optimization of the regression task because the Q-values can be equivalent to any real values. It is represented by equation (18),

$$\mathrm{L} = \frac{1}{2}\left[\mathrm{r} + \max_{a'} \mathrm{Q}(s', a') - \mathrm{Q}(s, a)\right]^2 \tag{18}$$

For every transition we replace the Q-table's update rule with these steps:

1) Fetch the Q-values that have been predicted for all possible actions with a feed-forward pass.
2) Compute the overall maximum of all the network outputs using function max (a,Q(s, a)).
3) Target the action's Q-value equal to sum(r,max (a,Q(s, a))) (make use of the computed max from the previous step). Set the values for all the other actions equal to initial values set in the Q-table in Step 1) this leads to a lesser error rate.
4) Weight Updation via backpropagation.

The Q-function is estimated using a Convolutional Neural Network (CNN) it is known as Convolution wrapped with Deep –Q(CONVDQN). But it turns out that approximation of Q-values using non-linear functions is not exactly stable. When the network is being trained, arbitrary small batches retrieved from storage are utilized rather than the most recent transition. This prevents over-fitting and due the next set of training samples being identical, that can cause the network to descend to a local minima. This work does not go into exploration-exploitation dilemma. It is to be noticed that whenever the Q-table is filled with arbitrary values, then the predictions are at first also arbitrary. After choosing a move having the largest value, the action chosen shall remain random and goes for rough exploration. As the function moves towards a single point, it returns more and more steady Q-values while the need of more of space exploration becomes less. So, basically, space-exploring is added as a part of the algorithm via Q-learning.

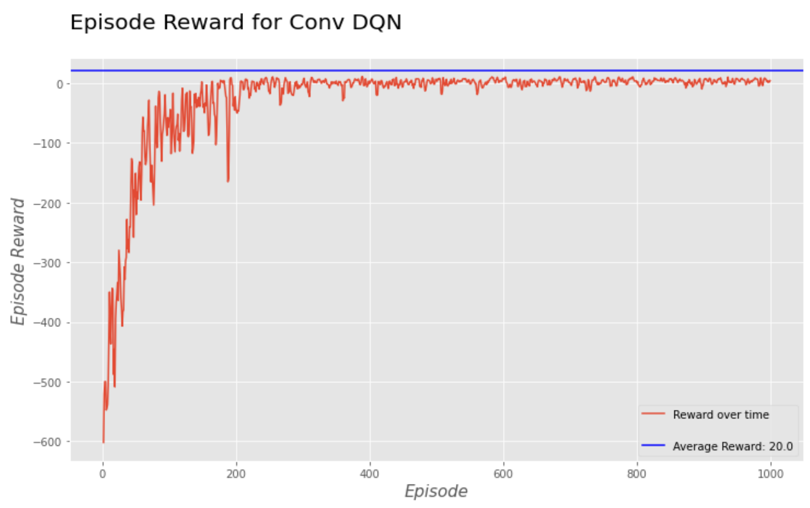

Fig. 6. Reward by ConvDQNs with respect to episode.

Fig. 6 shows the reward achieved by the ConvDQNs process. The rewards are measured by varying the number of episodes from 0 to 1000 with 200 intervals. The average reward achieved by ConvDQNs is 20.0.

cdqn, cdqn_stats =Conv_DQN()._learning(nenv = env3,num_episode = 1000,discount_factor(γ) = 0.8,alpha (α)= 0 .85, epsilon = 0.1,momentum = 0.95, aplha_decay = 0.01, hiddennode1 = 48, hiddennode2 = 24, batchsize = 32, epsilon_decay = 0.99995 ,epsilon_min = 0.01, state_size=1 , action_size=6.

4. Results and Discussion

In this part, performances of RL approaches using various metrics are given for utilizing the best performing model for future ATDs. To assess the model's efficacy, comparisons with ConvDQNs are presented in a variety of ways. The main evaluation metrics used were RMSEs (Root Mean Square Errors), MAEs (Mean Absolute Errors) Success Rates, and Wait Times.

RMSEs: Compute differences between expected and actual values. They concentrates on items with high discrepancies between anticipated and actual values, and smaller the disparity, better the algorithm’s performance. RMSEs can be defined using an equation (19),

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{19}$$

The predicted and actual values are indicated by $\hat{y}_i$and y_i, in similar way, and total measurements is expressed as n.

MAEs: They represent average absolute errors amongst predicted and observed values and focus on the sum of all differences between predicted and real values and defined as equation (20),

$$\mathrm{MAE} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}|\hat{y}_i - y_i|} \tag{20}$$

Fig 7 shows the performance comparison results of RMSE of 24 hours are measured between various methods. The proposed ConvDQNs shows lesser RMSE value of 4.77, whereas other methods such as NDRL, SARSA, QLN, and DQN gives increased RMSE value of 15.12, 11.92, 10.28, and 8.05 for 24 hours time period. It concludes that the proposed system has reaches the destination at correct time by agents. The agents can effectively finds the correct destination with lesser distance and speed manner. For all the time periods proposed system gives lesser RMSE value than the existing methods.

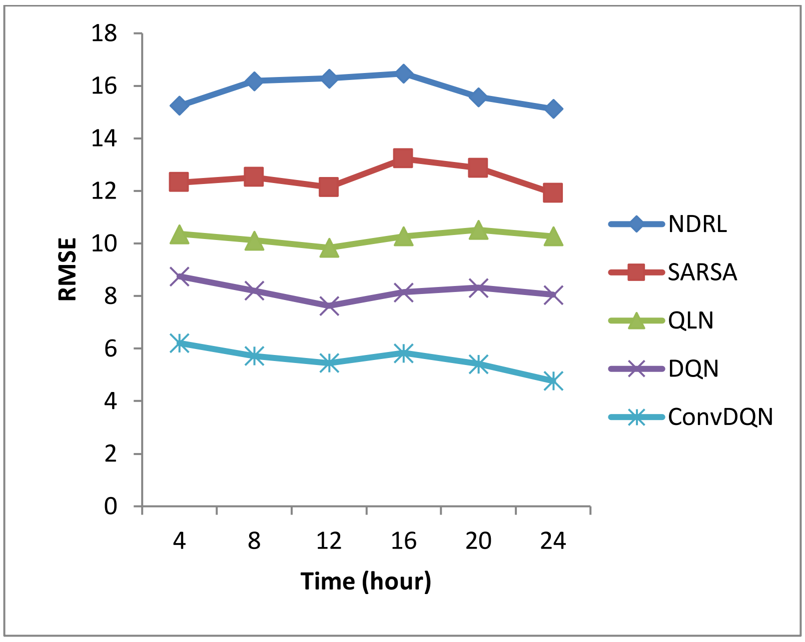

Fig. 7. RMSE comparison of methods under different time periods.

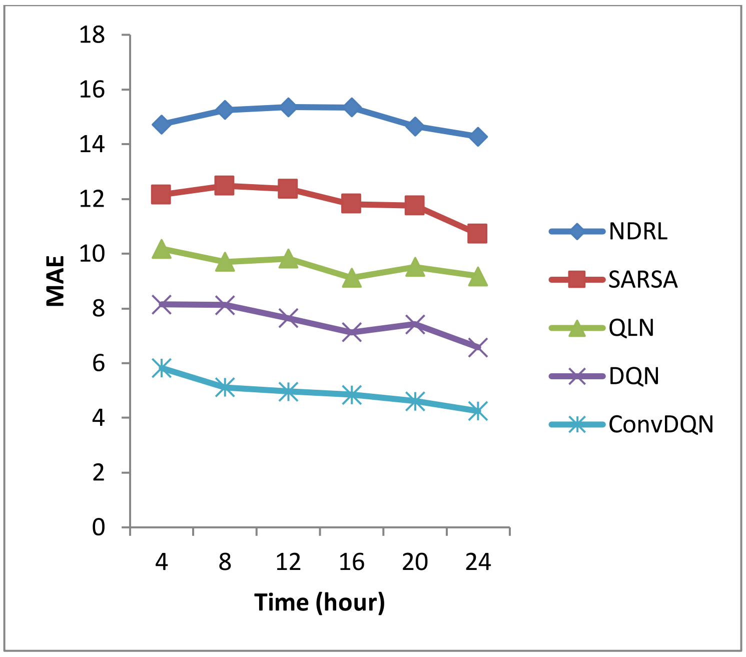

Fig. 8. MAE Comparison of methods under different time periods.

MAE results comparison of various methods with respect to different time periods are shown in the fig 8. The proposed system gives lesser MAE value of 4.25, whereas other methods such as NDRL, SARSA, QLN, and DQN gives increased MAE value of 14.28, 10.72, 9.18, and 6.58 for 24 hours time period. It concludes that the proposed system has reaches the destination at correct time by agents and correctly reaches the destination. It concludes that the proposed system guides to correct pickup and drop-off the passenger in correct time. For all time periods, proposed system gives lesser MAE value than the existing methods.

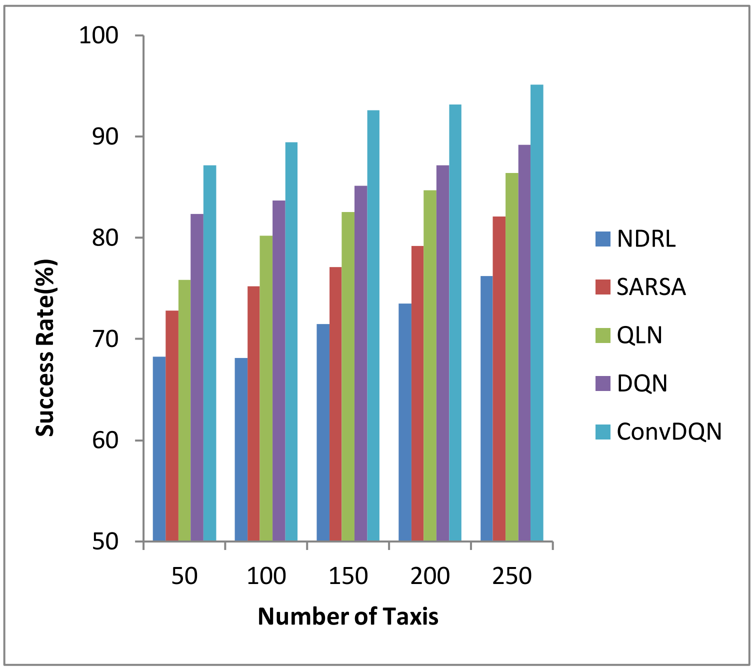

Fig. 9. Success rate comparison of methods under no. of taxis.

The success rate comparisons of various methods in terms of Taxis number are illustrated in figure 8. The proposed system gives higher success rate of 95.14%, whereas other methods such as NDRL, SARSA, QLN, and DQN gives lesser success rate of 76.25%, 82.14%, 86.41%, and 89.21% for 24 hours. Proposed system has correct time pickup and drop-off the passenger, it shows that the proposed system has higher success rate than the other methods.

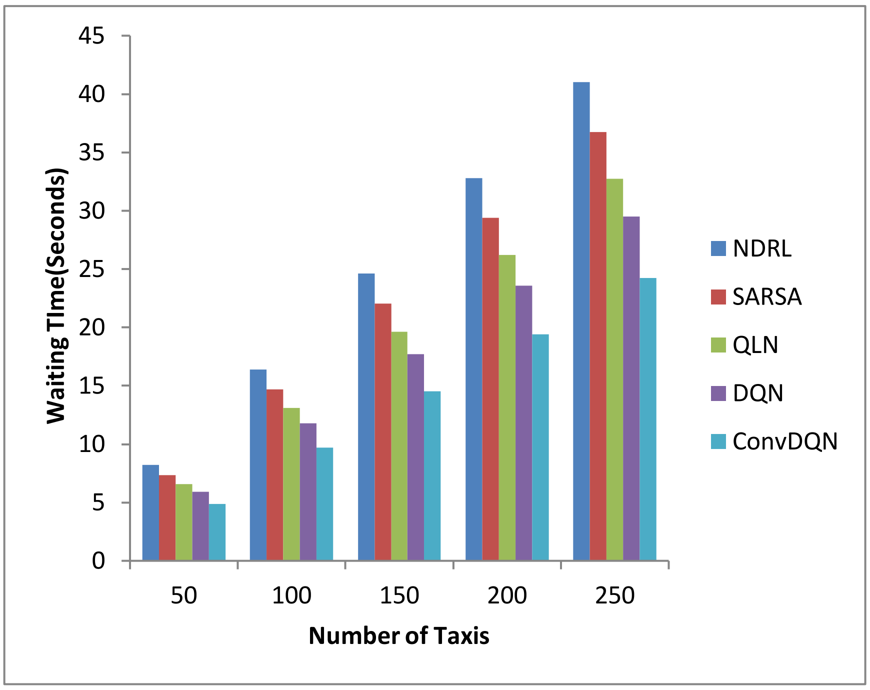

Fig.10. Wait times comparison of methods under no. of taxis.

The overall wait times comparison of various methods in terms of taxis' number is shown in figure 9. The proposed system has lesser wait times than the existing methods due to correct finding of the destination via agent helps in DQN. However the DQN alone doesn't improve the system efficiency in addition CNN is also used for correct approximation of Q value. The Q value only plays a major important role for giving suggestion to methods. The proposed system takes lesser wait times of 24.25 seconds, whereas other methods such as NDRL, SARSA, QLN, and DQN needs more wait times of 41 seconds, 36.75 seconds, 32.75 seconds, and 29.5 seconds for 24 hours.

5. Conclusion and Future Work

In this study, we provide an RL approach for ATDs that combines CNNs. First, as the MDPs progressed, the drivers' behaviours were constructed, taking into account the impacts after taking action in the long run. The state-action value function was calculated utilising the RL framework with MDPs and data expansion. In the ADL system, four RL approaches are introduced: Q-LNs , DQNs, SARSAs, and ConvDQNs. Through the use of agents and episodes, the ADL system can provide a safety mechanism for autonomous vehicles, enhancing vehicle safety and providing interpretability in the driving system. Driver scan determines the optimum decision and then quantifies the predicted future reward in a specific state using the value function in the ConvDQN system. Then, in order to anticipate taxi demand, four RL approaches are added to determine the proper pickup and drop-off location depending on passenger demands. Finally, the ADTs model based on agent RLs are utilised to transmit taxis between sites based on the present drop-off of taxis. The findings reveal that the ConvDQNs model outperforms the others on all four evaluation criteria, thus utilise the system's driving outcomes to simulate the real demand scenario in the passenger with on-time pickup and drop-off. The results confirmed that the new model performs better in increasing success rate, decreasing wait times , reducing MSEs (Mean Square Errors), and reducing RMSEs (Root Mean Square Errors) with relative percentages of 95.14 percent, 24.25 seconds, 4.25, and 4.77 for a 24-hour time period when compared to other benchmark methods. In the future, more fine-grained scheduling will be done; particularly, it will be studied which taxis should be sent in each grid, how to determine a route for each taxi, and where to discover customers once they arrive at the assigned grid. Solve these issues and increase the taxi service's efficiency.

References

[1] Sheng-Zu G. U., “Theoretical considerations and strategic choice on the development of smart city,” China Population Resources Environment, vol. 208, pp. 94–97, 2012.
[2] Neirotti P., A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, “Current trends in smart city initiatives: some stylised facts,” Cities, vol. 38, pp. 25–36, 2014.
[3] Hernafi, Y., Ahmed, M.B. and Bouhorma, M., 2016, An approaches' based on intelligent transportation systems to dissect driver behavior and smart mobility in smart city. In 2016 4th IEEE international colloquium on information science and technology (CiSt) ,pp. 886-895.
[4] Jin J., J. Gubbi, S. Marusic, and M. Palaniswami, “An information framework for creating a smart city through internet of things,” IEEE Internet of things Journal, vol. 1, no. 2, pp. 112–121, 2014.
[5] Zawieska, J. and Pieriegud, J., 2018. Smart city as a tool for sustainable mobility and transport decarbonisation. Transport Policy, 63, pp.39-50.
[6] Olszewski R., P. Pałka, and A. Turek, “Solving smart city transport problems by designing carpooling gamification schemes with multi-agent systems: the case of the so-called mordor of Warsaw,” Sensors, vol. 18, no. 2, pp.1-25, 2018.
[7] Kourti E., C. Christodoulou, L. Dimitriou, S. Christodoulou, and C. Antoniou, “Quantifying demand dynamics for Journal of Advanced Transportation 11 supporting optimal taxi services strategies,” Transportation Research Procedia, vol. 22, pp. 675–684, 2017.
[8] Vhaduri, Sudip, Christian Poellabauer, Aaron Striegel, Omar Lizardo, and David Hachen. "Discovering places of interest using sensor data from smartphones and wearables." In 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1-8, 2017.
[9] Dong W., L. Qian, X. Zhu, C. Jie, Y. Huang, and W. Chen, “Understanding travel behavior of private cars via trajectory big data analysis in urban environments,” in Proceedings of the 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 15th International Conference on Pervasive Intelligence and Computing, 3rd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress, Orlando, FL, USA, November 2017,pp. 917-924.
[10] Kyaw, T., Oo, N.N. and Zaw, W., 2018, Building travel speed estimation model for Yangon City from public transport trajectory data. In International Conference on Big Data Analysis and Deep Learning Applications, pp. 250-257.
[11] Dimitriou L., E. Kourti, C. Christodoulou, and V. Gkania, “Dynamic estimation of optimal dispatching locations for taxi services in mega-cities based on detailed GPS information,” IFAC-PapersOnLine, vol. 49, no. 3, pp. 197–202, 2016.
[12] Wei D., C. Yuan, H. Liu, D. Wu, and W. Kumfer, “The impact of service refusal to the supply demand equilibrium in the taxicab market,” Networks and Spatial Economics, vol. 17, no. 1, pp. 225–253, 2017.
[13] Rong H., Z. Wang, Z. Hui et al., “Mining efficient taxi operation strategies from large scale geo-location data,” IEEE Access, vol. 5, pp. 25623–25634, 2017.
[14] Kang C. and K. Qin, “Understanding operation behaviors of taxicabs in cities by matrix factorization,” Computers, Environment and Urban Systems, vol. 60, pp. 79–88, 2016.
[15] Zhang, W.; Honnappa, H.; Ukkusuri, S.V. Modeling urban taxi services with e-hailings: A queueing network approach. Transp. Res. Part C Emerg. Technol. 2020, 113, 332–349.
[16] Yuan, N.J.; Zheng, Y.; Zhang, L.; Xie, X. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. IEEE Trans. Knowl. Data Eng. 2013, 25, pp.2390–2403.
[17] Hwang, R.-H.; Hsueh, Y.-L.; Chen, Y.-T. An effective taxi recommender system based on a spatio-temporal factor analysis model. Inf. Sci. 2015, 314, pp.28–40.

[18] Luo, Z.; Lv, H.; Fang, F.; Zhao, Y.; Liu, Y.; Xiang, X.; Yuan, X. Dynamic Taxi Service Planning by Minimizing Cruising Distance Without Passengers. IEEE Access 2018, 6, pp.70005–70016.

[19] Ghosh, S.; Ghosh, S.K.; Buyya, R. MARIO: A spatio-temporal data mining framework on Google Cloud to explore mobility dynamics from taxi trajectories. J. Netw. Comput. Appl. 2020, 164, 102692.

[20] Ji, S.; Wang, Z.; Li, T.; Zheng, Y. Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning. Knowl. Based Syst. 2020, 205, pp.1-17.

[21] Musolino, G.; Rindone, C.; Vitetta, A. Passengers and freight mobility with electric vehicles: A methodology to plan green transport and logistic services near port areas. Transp. Res. Procedia 2019, 37, pp.393–400.

[22] Croce, A.I.; Musolino, G.; Rindone, C.; Vitetta, A. Sustainable mobility and energy resources: A quantitative assessment of transport services with electrical vehicles. Renew. Sustain. Energy Rev. 2019, 113, pp.1-13.

[23] Croce, A.I.; Musolino, G.; Rindone, C.; Vitetta, A. Transport System Models and Big Data: Zoning and Graph Building with Traditional Surveys, FCD and GIS. ISPRS Int. J. Geo-Inf. 2019, 8, pp.1-18.

[24] Gao, Y.; Jiang, D.; Xu, Y. Optimize taxi driving strategies based on reinforcement learning. Int. J. Geogr. Inf. Sci. 2018, 32, pp.1677–1696.

[25] Shou, Z.; Di, X.; Ye, J.; Zhu, H.; Zhang, H.; Hampshire, R. Optimal passenger-seeking policies on E-hailing platforms using Markov decision process and imitation learning. Transp. Res. Part C Emerg. Technol. 2020, 111, pp.91–113.

[26] Mao, C.; Liu, Y.; Shen, Z.J.M. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transp. Res. Part C Emerg. Technol. 2020, 115, pp.1-17.

[27] Wang, Z.; Qin, Z.; Tang, X.; Ye, J.; Zhu, H. Deep Reinforcement Learning with Knowledge Transfer for Online Rides Order Dispatching. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 617–626.

[28] Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Ke, W.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; Ye, J. Large-scale order dispatch in on-demand ridesharing platforms: A learning and planning approach. In Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018.

[29] Chen, J., Yuan, B. and Tomizuka, M., 2019, Model-free deep reinforcement learning for urban autonomous driving. In 2019 IEEE intelligent transportation systems conference (ITSC), pp. 2765-2771.

[30] Verma, T., Varakantham, P., Kraus, S. and Lau, H.C., 2017, Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 27, No. 1, pp. 409-417.

[31] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S. and Pérez, P., 2021. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, pp.1-18.

[32] Rasheed, I., Hu, F. and Zhang, L., 2020. Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN. Vehicular Communications, 26, pp.1-11.

[33] Jin, K., Wang, W., Hua, X. and Zhou, W., 2020. Reinforcement Learning for Optimizing Driving Policies on Cruising Taxis Services. Sustainability, 12(21), pp.1-19.

[34] Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Ke, W.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; Ye, J. Large-scale order dispatch in on-demand ridesharing platforms: A learning and planning approach. In Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018.

[35] Gao, Y.; Jiang, D.; Xu, Y. Optimize taxi driving strategies based on reinforcement learning. Int. J. Geogr. Inf. Sci. 2018, 32, 1677–1696.

[36] Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; Ye, J. A Deep Value-network Based Approach for Multi-Driver Order Dispatching. In Proceedings of the 24rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; Association for Computing Machinery (ACM), London, UK, 19–23 August 2018.

[37] Fan, J., Wang, Z., Xie, Y. and Yang, Z., 2020, A theoretical analysis of deep Q-learning. In Learning for Dynamics and Control (pp. 486-489). PMLR.

[38] Zhou, S., Liu, X., Xu, Y. and Guo, J., 2018, A deep Q-network (DQN) based path planning method for mobile robots. In 2018 IEEE International Conference on Information and Automation (ICIA) ,pp. 366-371.

[39] Yu, D., Ni, K. and Liu, Y., 2020. Deep Q-Network with Predictive State Models in Partially Observable Domains. Mathematical Problems in Engineering, vol.2020, no. 1596385, pp.1-9.

[40] Spano, S., Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Giardino, D., Matta, M., Nannarelli, A. and Re, M., 2019. An efficient hardware implementation of reinforcement learning: The q-learning algorithm. IEEE Access, 7, pp.186340-186351.

Authors' Profiles

Showkat A. Dar is a Research Scholar in the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University in Tamil Nadu India. He received his Bachelor of Engineering in Computer Science from Anna University Chennai, Tamil Nadu, India, and his Master of Engineering in Computer Science from Annamalai University, Tamil Nadu, India. He published seven papers published in various international journals. His research interests include image processing and computer vision.

Dr.S. Palanivel is Professor and Head in the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University Tamil Nadu India. He earned his Ph.D from Indian Institute of Technology Madras (IITM). He has 86 articles published in international journals, 26 papers published in international conferences, and 40 papers published in national journals and conferences. His research interests include Speech and Image processing, Machine and Deep Learning and other related fields.

Dr. M. Kalaiselvi Geetha is a Professor in the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University Tamil Nadu India. She received her Ph.D. in Computer Science and Engineering from Annamalai University and her M. Tech from the Indian Institute of Technology Delhi (IITD). She has 35 articles published in international journals, 10 papers published in international conferences, and 20 papers published in national journals and conferences. Her research interests include Artificial Intelligence, Video and Image Processing, Computer Vision, and other related fields.

How to cite this paper: Showkat A. Dar. S. Palanivel, M. Kalaiselvi Geetha, " Autonomous Taxi Driving Environment Using Reinforcement Learning Algorithms", International Journal of Modern Education and Computer Science(IJMECS), Vol.14, No.3, pp. 88-102, 2022.DOI: 10.5815/ijmecs.2022.03.06