A PRELIMINARY REPORT ON

A FEDERATED REINFORCEMENT LEARNING FRAMEWORK FOR AUTONOMOUS DRIVING

SUBMITTED TO

MR. K NAVEEN KUMAR
(PhD Research Scholar at IIT Hyderabad)

SUBMITTED BY

GAYATRI WALKE
SHUCHI TALATI
STUTI DHASMANA
VAIBHAVI KHARAPKAR
MKSSS's Cummins College of Engineering for Women, Pune
(B.Tech Computer Engineering)

ACKNOWLEDGEMENT

We would like to take this opportunity to express our sincere gratitude to everyone who has supported us throughout the course of this project on "**A Federated Reinforcement Learning Framework for Autonomous Driving**". We are thankful for their most generous guidance and constructive advice during the completion of this work.

We would also like to thank our Guide **Mr. K. Naveen Kumar** for supporting us throughout and giving his guidance. We are really grateful to him for indispensable support and suggestions.

We would also like to extend our gratitude to our College Guide **Dr. A. M. Naik** for giving us all the help and guidance we needed.

Gayatri Walke
Shuchi Talati
Stuti Dhasmana
Vaibhavi Kharapkar

MKSSS's Cummins College of Engineering for Women, Pune
(B.Tech Computer Engineering)

PROJECT ABSTRACT

Autonomous driving decision-making is a challenge owing to the complexity and uncertainty of the traffic environment. Reinforcement learning is considered to be a strong machine learning paradigm which can be used to train machines through interaction with the environment and learning from their mistakes. The Deep Deterministic Policy Gradient algorithm has great advantages in continuous control problems and plays a very important role in the field of autonomous driving. Federated Learning is a machine learning technique which trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. Federated Reinforcement Learning is an emerging area in the machine learning domain, and it already provides significant benefits over traditional, centralized machine learning approaches and has proven its significance in real-time applications as it provides results with low latency. The proposed Federated Reinforcement Learning framework will provide a solution to the challenges faced by the centralized machine learning approach in autonomous driving, for lane changing aspects. The models are trained faster as edge devices collaboratively train the model together. Moreover, it will enhance privacy and personalization of self-driving cars.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ABBREVIATION	ILLUSTRATION
RL	Reinforcement Learning
FL	Federated Learning
FRL	Federated Reinforcement Learning
DQN	Deep Q- Network
DDPG	Deep Deterministic Policy Gradient
TCP	Transmission Control Protocol
IID	Independent and Identically Distributed
TORCS	The Open Race Car Simulator
TD	Temporal Difference
IDE	Integrated Development Environment
UDP	User Datagram Protocol
FedAvg	Federated Average

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

1.1 Motivation

The desire to reclaim travel and commuting time, general convenience, and a way to reduce the stress and fatigue of driving for the consumers can be cited as the three primary motives behind working in the area of Autonomous Vehicles. This will give us a chance to explore several critical technologies that go behind safe and efficient autonomous-vehicle operations. While the short term goal will be to explore and implement the project the long term goal will always be that somewhere it leads to improving traffic safety and increasing access to travel.

1.2 Problem Definition

Most of the existing solutions focus on improving accuracy by training learnable models with centralized large-scale data that do not take users' privacy into account. To overcome the centralized learning challenge, we are developing a Federated Reinforcement Learning (FRL) framework for lane changing in autonomous vehicles. The framework will consist of the Deep Deterministic Policy Gradient algorithm (DDPG) for reinforcement learning and Federated Average algorithm (FedAvg) for Federated Learning. These two algorithms will be integrated and deployed on the TORCS, the 3D Open Racing Car Simulator.

Chapter 2: LITERATURE SURVEY

2.1 Background of domain

2.1.1 Federated Learning (FL)

Federated learning is a new machine learning technique that trains a global algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging data. This approach works in contrast to the traditional centralized machine learning techniques, in which all the local datasets are uploaded to one server, as well as to more classical decentralized approaches which assume that local data samples are identically distributed. For instance, mobile phones collectively study a shared prediction model, while keeping the device's training data local instead of sharing and storing it.

Federated learning processes involve training statistical models over remote devices or the siloed data centers, such as mobile phones or hospitals, while ensuring that data is kept localized. Training in heterogeneous and potentially massive networks brings novel challenges that require a fundamental departure from the standard approaches for large-scale machine learning algorithms, distributed optimization, and privacy-preserving data analysis.

2.1.2 Reinforcement Learning (RL)

Reinforcement learning (RL) is an important area of machine learning that is concerned with how intelligent agents ought to take desired actions in an environment to maximize the notion of cumulative reward. It differs from supervised learning in not needing labeled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the main

focus is on finding a balance between exploration (exploration of uncharted territory) and exploitation (exploitation of current knowledge).

The primary purpose of reinforcement learning is for the agent to learn an optimal, or nearly-optimal, policy that can maximize the "reward function" or other user-provided reinforcement signals that gets accumulated from the immediate rewards.

Reinforcement learning is well studied in many disciplines, such as statistics, game theory, operations research, control theory, simulation-based optimization, information theory, multi-agent systems, and swarm intelligence.

2.2 Comparisons, Research Paper Studied

- **Research Paper 1:**
    - **Citation:** C. Nadiger, A. Kumar and S. Abdelhak, "**Federated Reinforcement Learning for Fast Personalization**," IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2019, pp. 123-127, doi: 10.1109/AIKE.2019.00031 (2019)
    - **Approach:** Agent is a DQN Model. The DQN training involved two neural networks namely Q and Target Q network
    - **Results/Accuracy:** The results show the efficacy of the approach by testing it on 3, 4, and 5 agents, where the speed up of the personalization time is ~ 17%.
    - **Key Takeaway:** Overall architecture for federated reinforcement learning (FRL), which primarily includes the grouping policy, the learning policy, and the federation policy
    - **Future Scope:** Extending the approach to areas where fast personalization in a changing environment is needed

- **Domain:** Federated learning, reinforcement learning; computer games, game personalization

- **Research Paper 2:**
  - **Citation:** Loiacono, Daniele, Pier Luca Lanzi Alessandro, Prete and Luigi Cardamone. "**Learning to overtake in TORCS using simple reinforcement learning.**" IEEE Congress on Evolutionary Computation (2010)
  - **Approach:** Reinforcement learning : Q learning with Behavior Analysis and Training (BAT) methodology
  - **Results/Accuracy:** Results show that even simple Q Learning can produce competitive overtaking behaviors by 90.16% Success Rate
  - **Key Takeaway:** Reinforcement learning in TORCS - Reward function, Q table

    TORCS sensors, car state usage for programming
  - **Future Scope:** Integrating neural networks to implement Q learning to increase efficiency
  - **Domain:** Autonomous Driving, TORCS

- **Research Paper 3:**
  - **Citation:** L. Yi, "**Lane Change of Vehicles Based on DQN**," 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), pp. 593-597, doi: 10.1109/ISCTT51595.2020.00113 (2020)
  - **Approach:** DQN, MAXMIN Q-Learning
  - **Results/Accuracy:** Utilization of Q-learning to change lanes successfully validated on simulator Deep-trafic
  - **Key Takeaway:** DQN advantages over Q- learning for autonomous driving- reduce training time and storage space. Different options for RL actions in the simulator.

- **Future Scope:** Multi step lane changing, real time application
- **Domain:** DQN, Autonomous Driving

- **Research Paper 4:**
  - **Citation:** Vitelli, M. Nayebi, A.: CARMA: "**A deep reinforcement learning approach to autonomous driving**". Technical report Stanford University, Tech. Rep., (2016)
  - **Approach:** Comparative analysis of different reinforcement learning methods to perform the task of autonomous car driving from raw sensory inputs
  - **Results/Accuracy:** Strongest agent seemed to be the discrete Q-learning agent. While the agent's average reward was much lower than both the greedy agent and the DQN agent, its average speeds were much higher.
  - **Key Takeaway:** Deep Q-Networks can be an effective means of controlling a vehicle directly from high-dimensional sensory inputs
  - **Future Scope:** A better alternative of defining their reward function that still maintains the careful balance between maximizing speed while guaranteeing car stability
  - **Domain:** Deep Q-network (DQN), Autonomous Driving

- **Research Paper 5:**
  - **Citation:** T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "**Federated Learning: Challenges, Methods, and Future Directions**," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, May 2020
  - **Approach:** Core challenges in FL, privacy preservation, survey of current work and future directions in FL
  - **Results/Accuracy:** System heterogeneity might cause the network. Privacy prevention comes at the cost of reduced system performance and efficiency

- **Key Takeaway:** Communication-efficiency, system heterogeneity, data heterogeneity, and privacy are the major challenges associated with Federated learning
- **Future Scope:** To find a robust methodology to overcome these major challenges in Federated learning
- **Domain:** Federated Learning

- **Research Paper 6:**
  - **Citation:** Pérez-Gil, Óscar & Barea, Rafael & López-Guillén, Elena & Bergasa, Luis & Revenga, Pedro & Gutierrez, Rodrigo & Díaz, Alejandro. ."**DQN-Based Deep Reinforcement Learning for Autonomous Driving**", book Advances in Physical Agents II
  - **Approach:** DQN-CNN DQN-FC DQN-FLATTEN -IMAGE DQN-PILOTNET DQN-CNN, DQN-FC, DQN-FLATTEN -IMAGE DQN-PILOTNET
  - **Results/Accuracy:** All agents reach the final destination in the established episodes. PilotNet is a good practical solution for application All agents reach the final destination in the established episodes. PilotNet is a good practical solution for application
  - **Key Takeaway:** Comparison of Deep Reinforcement Learning Agents. Approach for RL in CARLA similar to TORCS. Comparison of Deep Reinforcement Learning Agents
  - Approach for RL in CARLA similar to TORCS.
  - **Future Scope:** Reduce training time. Reduce training time
  - **Domain:** Autonomous Driving, Reinforcement Learning Autonomous Driving, Reinforcement Learning

Chapter 3: REQUIREMENTS

3.1 Background of Domain

However, Reinforcement Learning and Deep Reinforcement Learning still face a number of important technical and non-technical challenges in solving real-world problems such as autonomous driving in spite of their excellent performance in many areas. Due to the successful application of FL to supervised learning tasks, there is interest in investigating similar approaches in RL, namely FRL. In contrast, FL is useful in certain specific situations, but fails to cope with cooperative control and optimal decision-making in dynamic environments.

The FRL not only allows agents to learn to make good decisions in an unknown environment, but it also ensures that all data collected by the agents during their exploration is not shared with others. Thus, we propose using the FL framework to enhance the security of RL and define FRL as a security-enhanced distributed RL framework that accelerates the learning process, protects agent privacy and handles Not Independent and Identically Distributed (NIID) data. In addition to enhancing RL's security and privacy, we believe that FRL has significant potential to help RL achieve better performance in various areas, which are described in the following subsections.

3.2 Software Requirement Specifications (SRS)

3.2.1 Scope of the Project

The scope of the project includes the following:

- Development of a Federated Learning(FL) framework to overcome statistical heterogeneity

- Designing a FRL algo for accurate lane changing behavior such as:
- monitoring vehicle tracks
- 3. Exploring TORCS Simulator to deploy the developed FRL Model.
- 4. Integration of individual models to generate a unified FRL framework for lane changing in autonomous vehicles

The scope of this project does not include:

- Focusing on multiple clients, single vehicle case i.e. considering the same type of models for all vehicles
- Developing or designing any new tracks, only using ones available in the TORCS

3.2.2 Functional Requirements

- Make use of sensors such as, speed, angle, fuel, etc. to control TORCS Actions in order to keep the client in its lane while driving the car

- Prevent client from stopping in certain positions on the track or moving out of the track

3.2.3 Interface Requirements

- Client connects with server through the socket

- Server and client communicates with each other with help of a TCP connection

3.2.4 Design Constraints

- Deployment only on stimulator

- Use of already available TORC Simulator tracks

3.2.5 Non Functional Attributes

- **Security:** The framework provides a secure way to share weights of locally trained model at the client side with the global model with help of cryptographic methods

- **Scalability:** Simulation makes large-scale testing of autonomous vehicles practically possible

- **Portability:** The model can be accessed by clients only inside the given simulator

- **Reliability:** Reduces accidents caused due to human error, which make up to 94 percent of the total serious crashes.

- **Reusability:** The trained model can be re-used by the clients till a new model is received from the server

3.2.6 Use Case Diagram

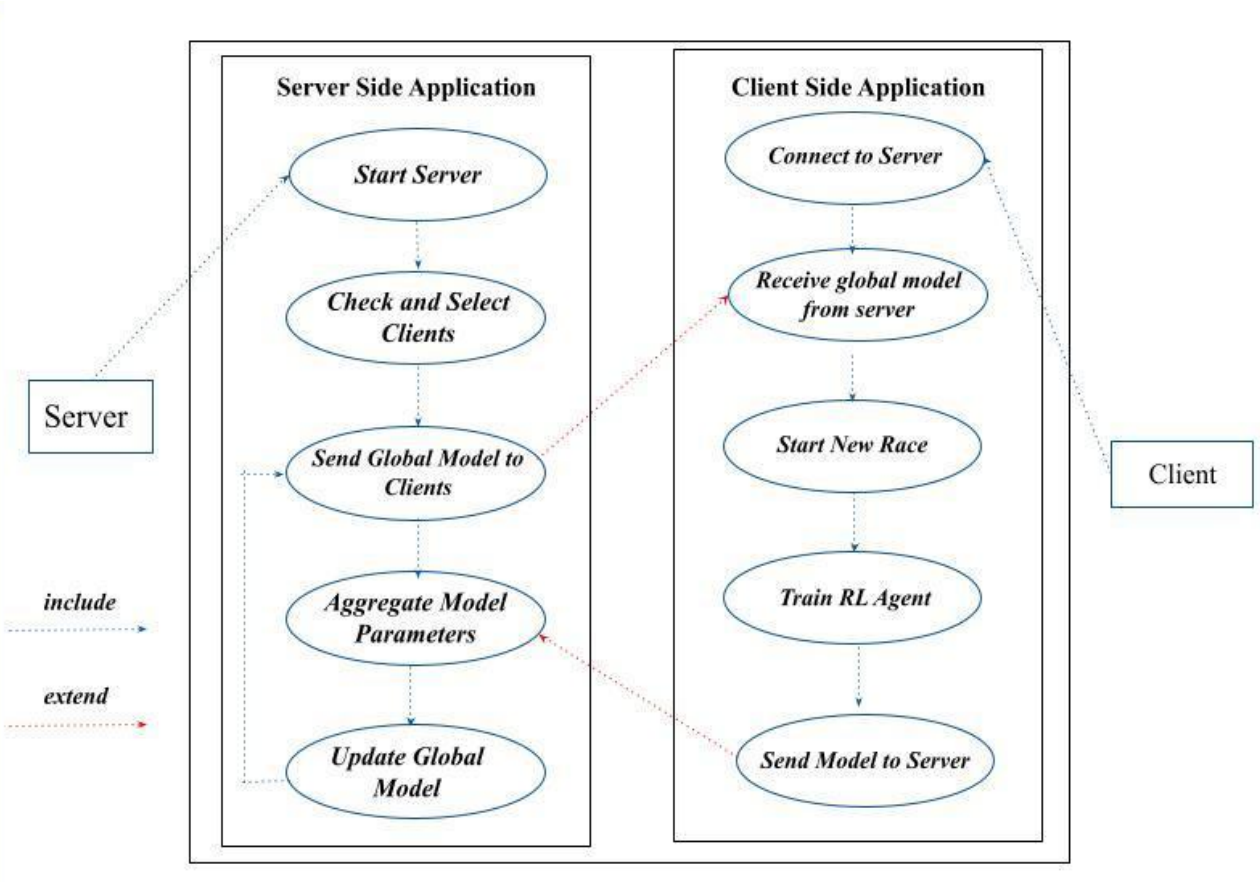

Figure 3.2 Use Case Diagram

Chapter 4: SIMULATION ENVIRONMENT

4.1 The Open Racing Car Simulator (TORCS)

The open racing car simulator (TORCS), is a modern, modular, highly portable multi-player, multi-agent car simulator. Its high degree of modularity and portability render it ideal for artificial intelligence research. It provides a sophisticated physics engine, 3D graphics, and several diverse tracks and car models. The car has access to all information of the environment and the server acts as a proxy for the environment.

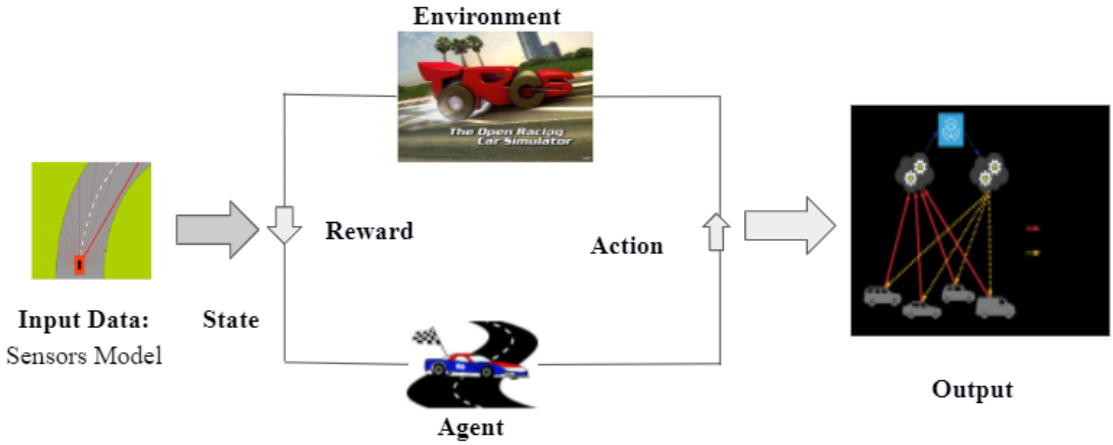

Figure 4.1 TORCS Environment

4.2 TORCS Sensors(Input)

- ***Angle:*** Angle between the car direction and the direction of the track axis
- ***CurLapTime:*** Time elapsed during current lap
- ***Damage:*** Current damage of the car (the higher is the value the higher is the damage)
- ***distFromStartLine:*** Distance of the car from the start line along the track line
- ***distRaced:*** Distance covered by the car from the beginning of the race
- ***Fuel:*** Current fuel level
- ***Gear:*** Current gear: -1 is reverse 0 is neutral and the gear from 1 to 6

- ***lastLapTime:*** Time to complete last lap
- ***Rpm:*** Number of rotations per minute of the car engine
- ***speedX:*** Speed of the car along the longitudinal axis of the car
- ***speedY:*** Speed of the car along the transverse axis of the car
- ***Track:*** Vector of 19 range finder sensors: each sensor represents the distance between the track edge and the car
- ***trackPos:*** Distance between the car and the track axis
- ***wheelSpinVel:*** Vector of 4 sensors representing the rotation speed of the wheels

4.3 TORCS Control Actions(Output)

- ***Accel:*** Virtual gas pedal (0 means no gas, 1 full gas)
- ***Brake:*** Virtual brake pedal (-1 means no brake, 1 full brake)
- ***Gear:*** Gear value
- ***Steering:*** Steering value: -1 and +1 means respectively full left and right, that corresponds to an angle of 0.785398 rad
- ***Meta:*** This is meta-control command: 0 Do nothing, 1 ask server to restart the race

Chapter 5: SYSTEM DESIGN

5.1 System Architecture

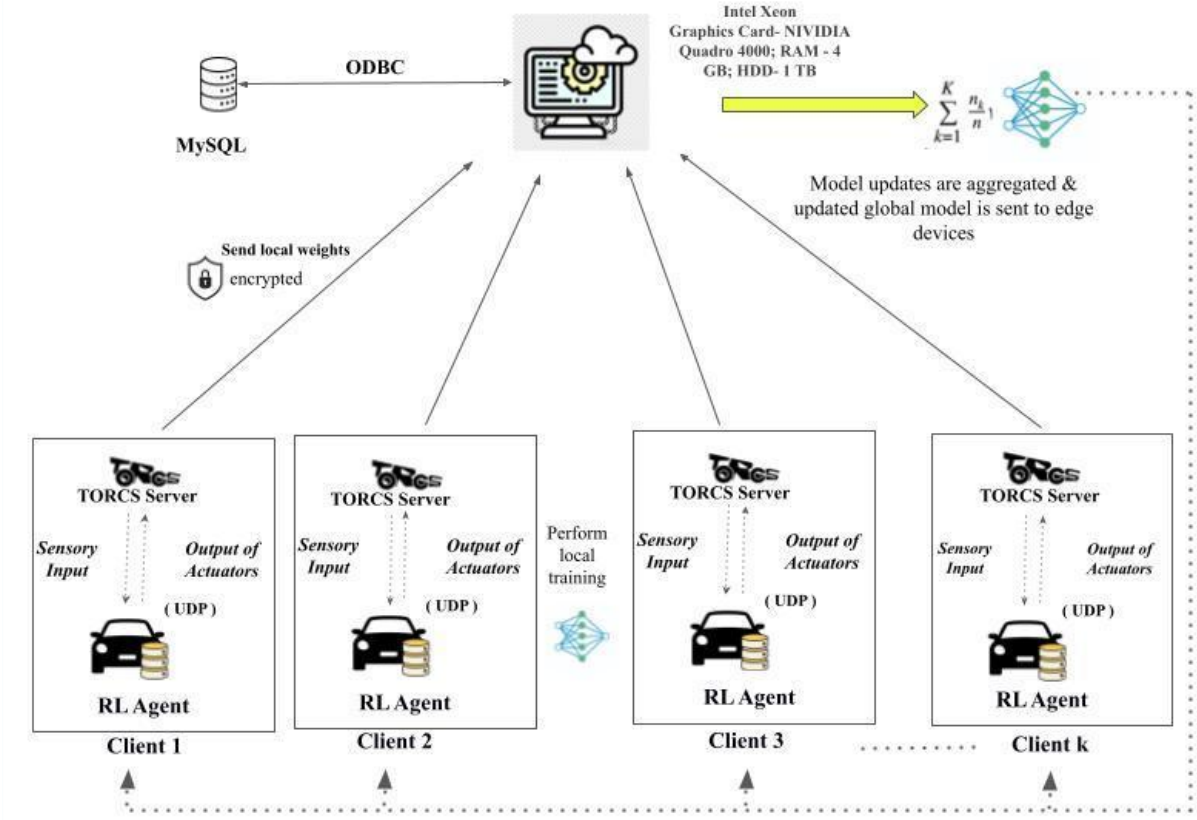

Figure 5.1 System Architecture

5.2 Algorithms

5.2.1 Q - Learning

- Q-Learning is a basic form of Reinforcement Learning. It uses Q-values to iteratively improve the behavior/actions of the learning agent. Q-values are also known as the action values.

- It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."

5.2.1.1 Q - Learning Algorithm

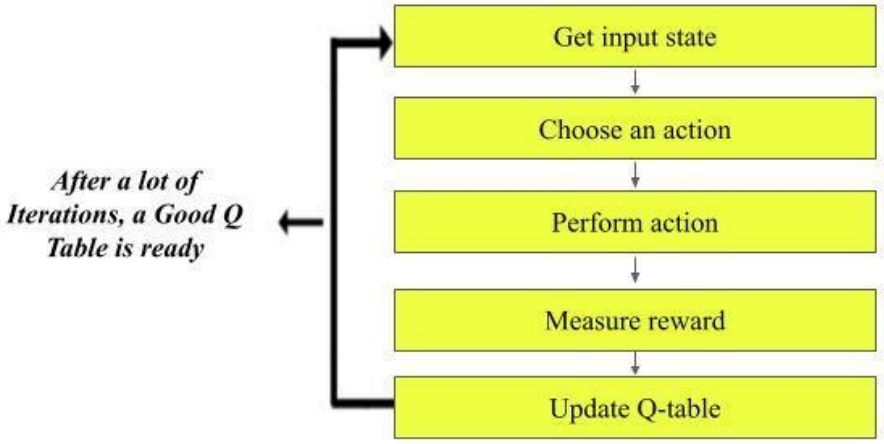

Figure 5.2 Q-Learning Algorithm

5. 2.1.2 Input States of Q - Learning Algorithm

- ***speedX:*** The speed of the car along its longitudinal axis.
- ***Angle:*** Angle between the direction of the car and the track axis.
- ***Track pos:*** Distance between the car and the track axis
- ***Track:*** Distance sensor at −40◦, −20◦, 0◦, 20◦, 40◦ Distance between the car and track [5, 7, 9, 11]

5. 2.1.3 Action States of Q - Learning Algorithm

- ***Accel:*** Virtual gas pedal (0 means no gas, 1 full gas)
- ***Brake:*** Virtual brake pedal (-1 means no brake, 1 full brake)

- ***Gear:*** Gear value
- ***Steering:*** A steering value of -1 or +1 corresponds to full left and right, giving an angle of 0.785398 radians.
- ***Meta:*** This is meta-control command: 0 Do nothing, 1 - restart.

5.2.1.4 Rewards in Q - Learning Algorithm

- Car make good lane keeping
    - the reward will be positive and high if the distance was long and in general it has a continuous range from [-1,1]
- Stop in certain position
    - the reward will be negative and equal to -1
- The car goes out of the track
    - the reward will be negative and equal to -1 and we will restart the race

5.2.2 Deep Deterministic Policy Gradient (DDPG)

It is a model-free off-policy algorithm for learning continuous actions. It is the combination of ideas from Deterministic Policy Gradient (DPG) and Deep Q-Network (DQN). It is based on DPG, which can operate over continuous action spaces, and it utilizes Experience Replay and slow-learning target networks from DQN.

Combination of 3 techniques:

1. Deterministic Policy-Gradient Algorithms
2. Actor-Critic Method
3. Deep Q-Network

In DDPG algorithm topology consists of two copies of network weights for each network, (Actor: regular and target) and (Critic: regular and target).

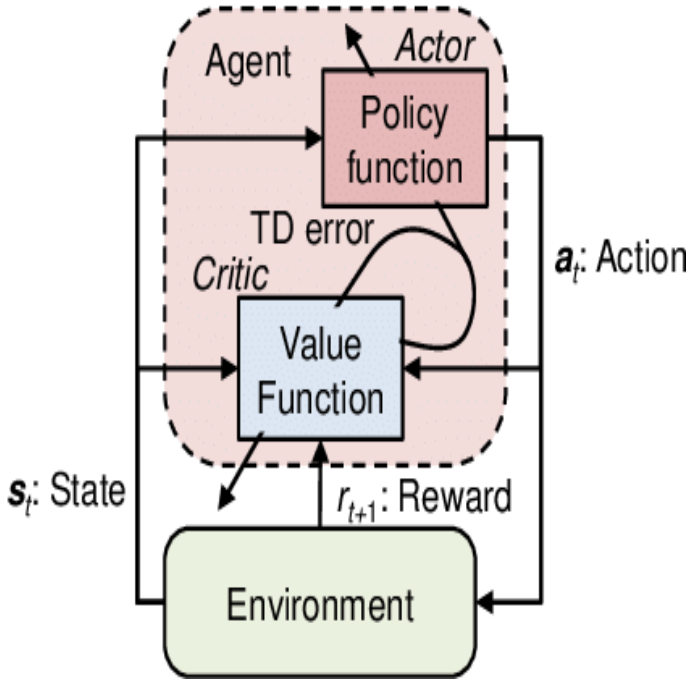

Figure 5.3 Actor Critic Network

5.2.3 DDPG Algorithm

1. The **Actor network** which takes input as a state ‘s’ and results in the action ‘a’.
2. The **Critic network** which takes an input as a state s and action a and returns the Q value.
3. Define a **target network** for both the Actor network and Critic network respectively.
4. Perform the update of Actor network weights with **policy gradients** and a Critic network weight with the gradients calculated from the **Temporal Difference** (TD) error.
5. Add an **exploration noise** N to the action produced by the Actor.

6. Selected action in a state, s, receive a reward, r and move to a new state, s'.
7. Store this transition information in an experience **replay buffer**.
8. Train the network, and then we calculate the **target Q value**. Compute the **TD error**. Update of the Critic networks weights with gradients calculated from the error.
9. Update our **policy network** weights using a policy gradient.
10. Update the weights of Actor and Critic network in the **target network**.

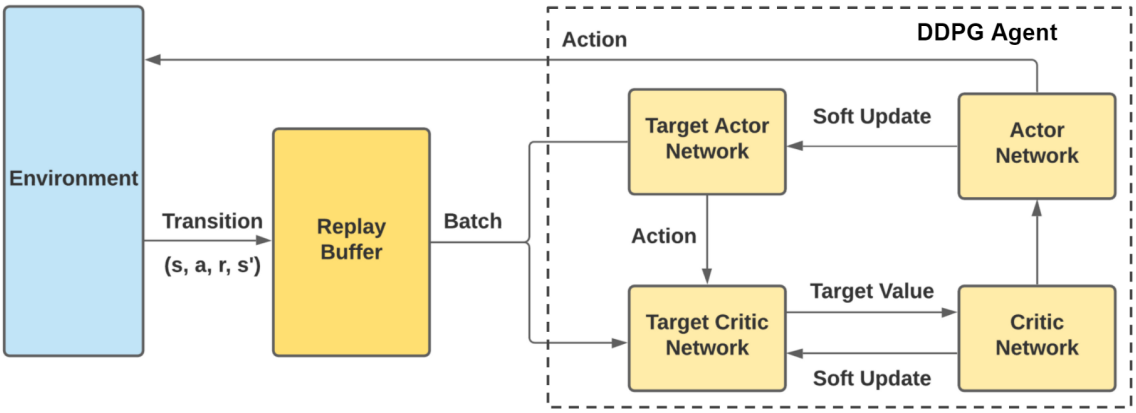

Figure 5.4 Flow of DDPG

5.2.4 FedAvg - Algorithm for Federated Learning

- A central parameter server is used to communicate between the clients in Federated Averaging (FedAvg) and the client's data is kept local for privacy protection.
- Average Function is for model updates aggregation
- A model is trained in a distributed manner where each of the clients trains the model locally.

5.2.5 FedAvg - Algorithm for Federated Learning

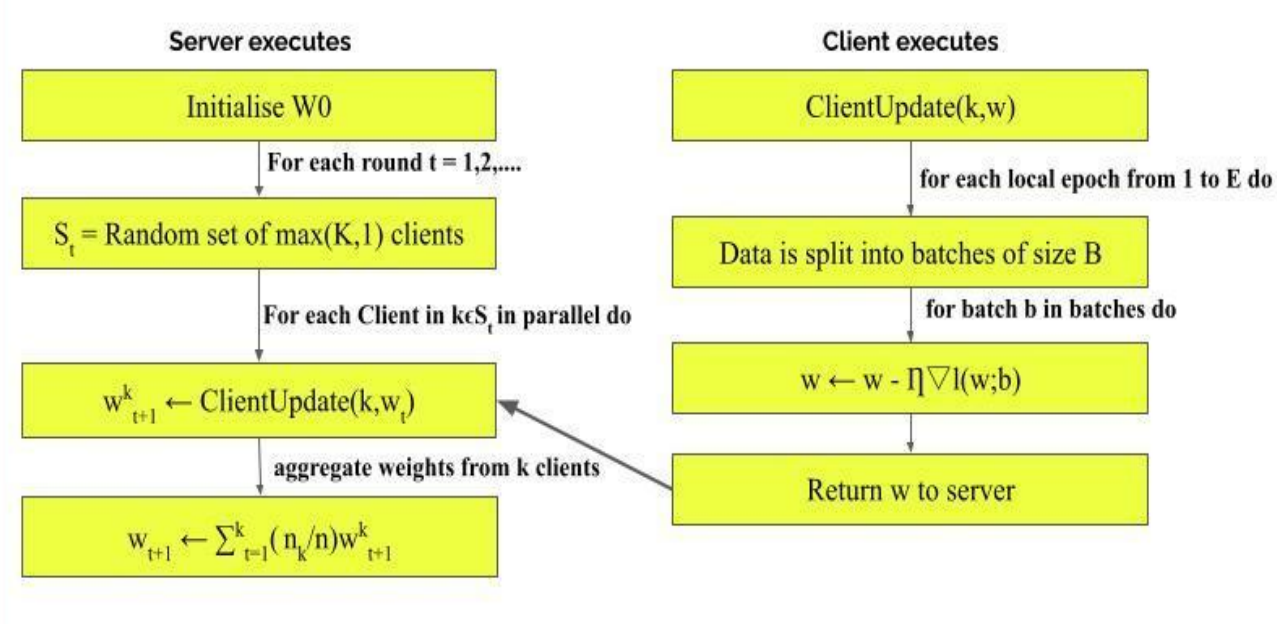

Figure 5.5 FedAvg Algorithm

Above is the flowchart for FedAvg Algorithm where the clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Aggregation is done in a weighted averaging manner where clients with more data influence more significantly the newly aggregated model.
The obtained model is sent back to all the clients, and another communication round can start. Below is the Aggregation Function used for FedAvg Algorithm.

$$Min\ F\ (w),\ where\ F\ (w) := \sum_{k=1\ to\ m} p_k F_k\ (w\ )$$

Here m is the total number of devices, F_k is the local objective function for the k^{th} device, and p_k specifies the relative impact of each device with $p_k >= 0$.

After the selection of K number of clients, few clients might backout, in such cases the server waits for a few instances of time before dropping them out using a timeout resulting in straggling mitigation and providing personalization.

Chapter 6: TECHNOLOGY

Operating System: Windows/Linux

Simulator Environment: TORC 3D Simulator

IDE: Jupyter Notebook, PyCharm

Programming Language: Python

API: WebSocket

Libraries:

1. **pytorch**: provides a collection of workflows to develop and train the RL models
2. **tensorflow**: provides a collection of workflows to develop and train FL and RL models
3. **tensorFlow federated**: a federated learning framework that provides computations and algorithms for FL
4. **Socket**: connects the server host given a specific port and client.
5. **Pandas**: to structure the data (for importing and analyzing data)
6. **Numpy**: to perform mathematical and logical operations on arrays
7. ***gymTorcs:*** provides connection between TORCS server and code

Chapter 7: IMPLEMENTATION ASPECTS

7.1 Q-Learning Implementation

7.1.1 Functions

1. **carControl**

It has the Control functions that allow you to control a car in a game. You give it values, then it parses them and sends them to the server to move the car.

2. **carState**

This function provides the values describing the state of the car.A car's state is described by its sensor values, its distance from the start, its damage, etc.

3. **msgParser**

It provides a UDP message receiver and builder for server-client communication. It constructs the messages for the control actions and sends them to the server. Further, it accepts the UDP messages from the server that represents the car state.

4. **pyclient**

It is the client code that connects to the server host at specified port and socket. Subsequently it calls the driver function (which is our game loop) and then transfers the UDP messages to the server using the msgParser.

7.1.2 Drive implementation

1. CheckStuck Function

This function checks whether the car is stuck if the car angle exceeds 45 degrees, it is considered stuck. If it stays stuck for more than 25 game ticks and the traveled distance is less than 0.01m, an episode ends. Every time the agent is not stuck, the stuck timer resets to zero.

2. Learning Interface

The primary functions of the learning interface are: Action Selection and updating of learning algorithms. The learning interface consists of 4 functions: GetState, RewardFunction, QtableUpdate, ActionSelection.

7.1.3 Learning Interface Steps

1. GetState Function

Firstly, the speed and distance sensor values are discretized as follows:

- speedList = [0,10,20,30,40,60,70,80…200]
- distList = [-1,0,5,10,20,30,40,50..200]
  These values carefully cover all possible states. As they are 16 discretized values, each of them is represented in 4 bits of binary form.
  If only one of the 5 sensors are used, as representing all 5 sensors would take 20 bits, which would mean 220 states, which is not feasible.
  Hence, we used another 3 bits to distinguish between the maximum of the 5 sensors which gives us a total of 7 bits describing the sensor values.
  So finally we get a total of 11 bits for the states which corresponds to 2048 possible states.

2. ActionSelection Function

Discretized action values are as follows:

Steer	Accelerate(1)	Neutral(0)	Brake(-1)
0.5(left)	0	1	2
0.1(left)	3	4	5
0	6	7	8
-0.1(right)	9	10	11
-0.5(right)	12	13	14

Table 7.1 Action Selection

An action selection is based on a random number. This number can be between 0 and 1 Policy(s) = Informed if num < eta Random if num < eta+epsilon Max action otherwise.

Each time step, there is a probability that a heuristic action will be taken, a random action or a greedy action will be taken.

Random exploration is necessary to learn to improve upon the heuristic policy. The highest Q-value in the Q-table given the current state is the Max Action.

3. RewardFunction

There can be three different scenarios for Reward Function:

- If the car is stuck

-2 is taken as a reward and sends a meta action for a re-start. This was an undesirable action, and hence it is ensured that it is never repeated.

- If the car is out of track (abs(track position) > 1)

It takes -1 as a reward

- If the car is neither stuck or out of track

Depending on the track position, angle, and distance traveled, a reward is taken, with a maximum value of 1.

4. QtableUpdate

The state already exists in the table or not is checked first. If the state does not exist, we create it in the Q-table, which is already initialized with zero for all actions.

After that, according to the current state, the previous state, action and reward, the Q-values of the previous state are updated.

There is also a possibility for multiple actions to have the same value, like when a new state is explored and all values are unknown (and all have default value 0).

7.2 Deep Deterministic Policy Gradient (DDPG) Implementation

7.2.1 Actions

- ***Acceleration:*** Virtual gas pedal (0 means no gas, 1 full gas)
- ***Brake:*** Virtual brake pedal (-1 means no brake, 1 full brake)
- ***Steering:*** Steering value: -1 and +1 means respectively full left and right

7.2.2 Reward Function

The reward function is as stated below:

$$R_t = V_x cos(\theta) - V_x sin(\theta) - V_x \mid trackPos \mid$$

7.2.3 Termination Judgment

Episode terminates if:

- the progress of agent is small
- the agent runs backward
- the car is out of track

The reward will be equal to -200 when the race is restarted.

7.2.4 Collision Detection

If the damage is more than Previous Action damage, then 1 is subtracted from the Reward.

7.2.5 Exploration policy

The Ornstein-Uhlenbeck (OU) Process is used for the exploration policy instead of a greedy approach, It is a stochastic process which has mean-reverting properties.

- θ : how “fast” the variable reverts towards to the mean
- μ : equilibrium or mean value
- σ : degree of volatility of the process

The following table shows the values for action:

Action	θ	μ	σ
steering	0.6	0.0	0.30
acceleration	1.0	[0.3-0.6]	0.10
brake	1.0	-0.1	0.05

Table 7.2 Exploration Policy

Chapter 8: RESULTS

Figure 8.1 Q leaning performance on Aalborg Track

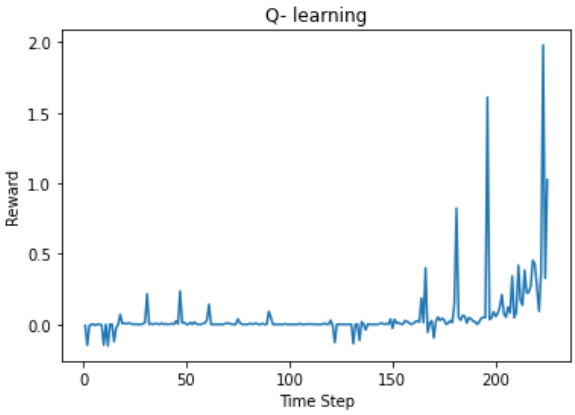

Figure 8.2 Graph for Reward vs Time Step for Q-Learning

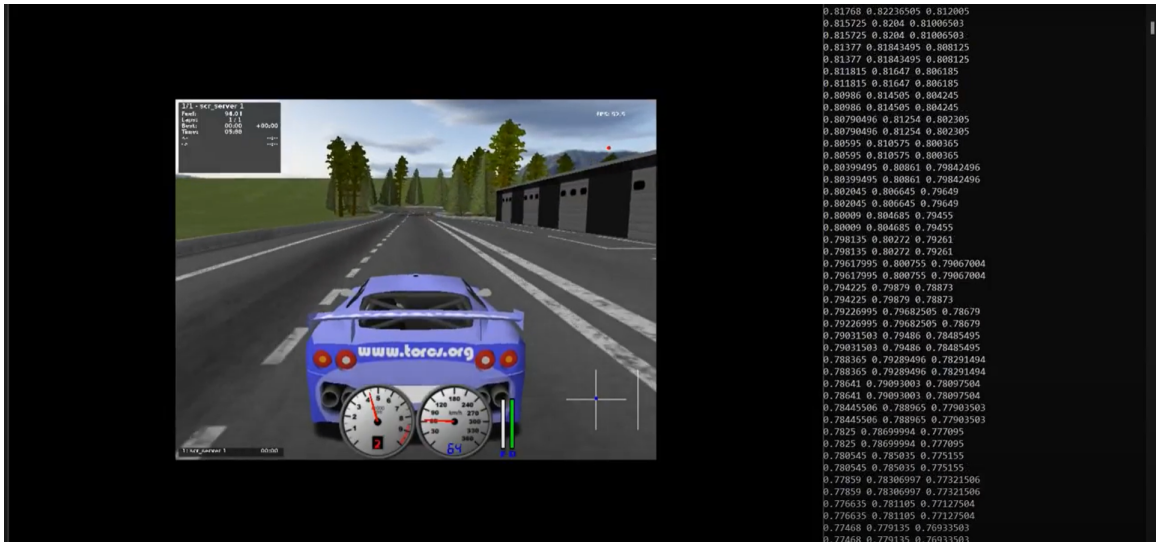

Figure 8.3 DDPG performance on Aalborg Track

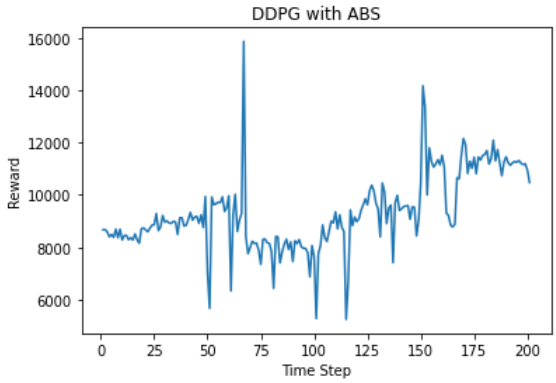

Figure 8.4 Graph for Reward vs Time Step for DDPG

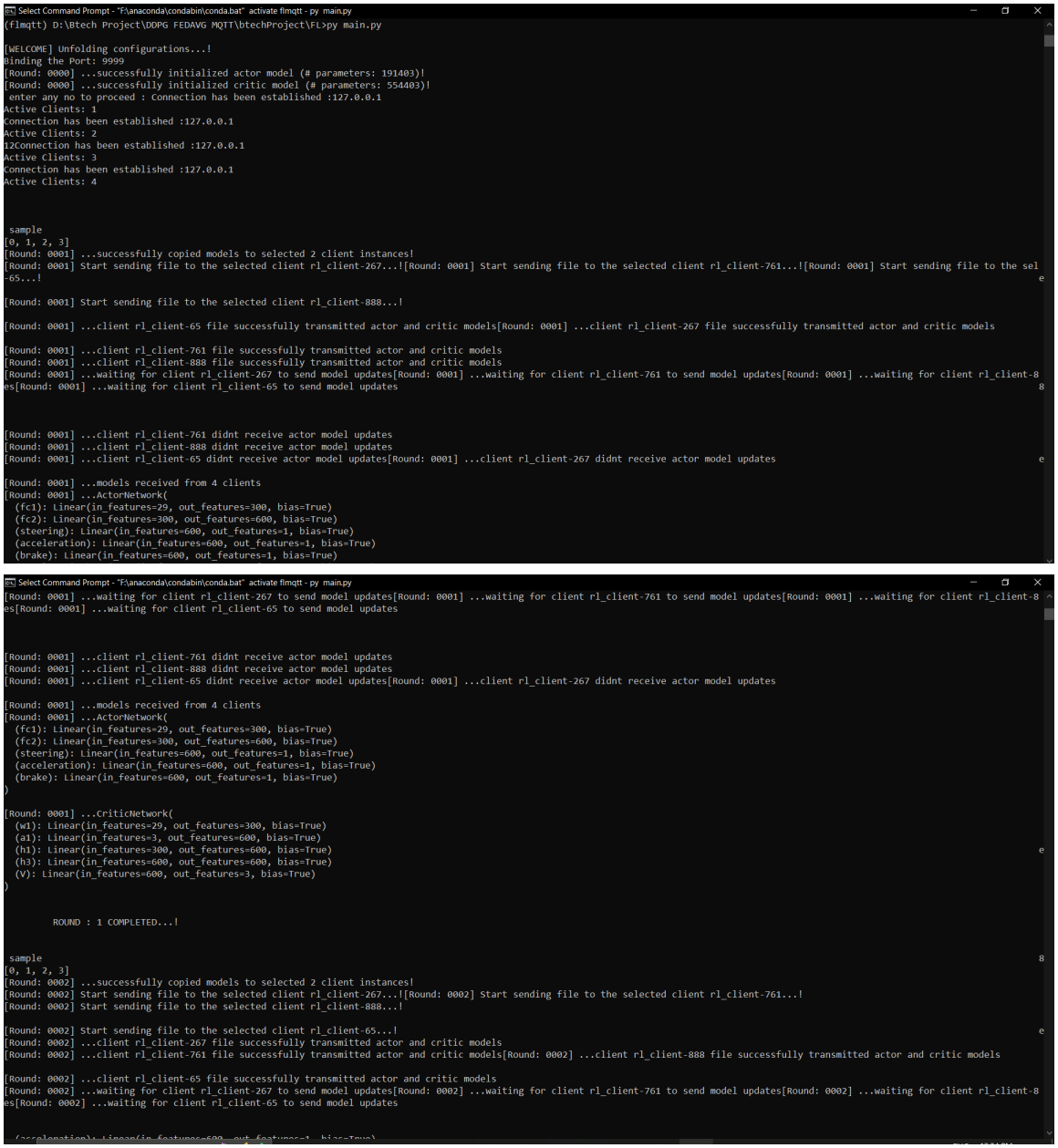

Figure 8.5 FRL Aggregation Performance on Server side

Figure 8.6 FRL Aggregation Performance on Aalborg Track-Client side

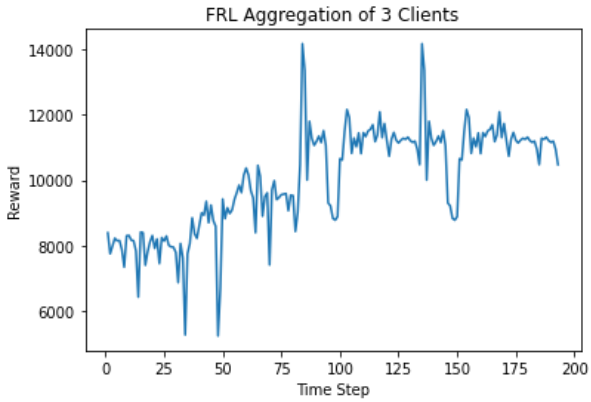

Figure 8.7 Graph for Reward vs Time Step for FRL Aggregation

Concept	Reward
DDPG	9894.30
FRL	11473.70
Percent increase	**15.96 %**

Table 8.1: Average values of reward

As can be seen from Table I, the average reward value for DDPG alone is 9894.3, whereas for FRL the same turns out to be 11473.7, indicating an increase of approximately 16%. The results indicate that the FRL framework can accelerate the training speed and improve the performance of the federation of three cars.

Chapter 9: CONCLUSION AND FUTURE WORK

9.1 Conclusion

- From results obtained above, it can be concluded that the policy-based methods offer better convergence properties, and are more effective in high dimensional action spaces when using continuous actions as compared to value based approaches.
- The proposed FRL framework can accelerate the training speed and improve the performance of the federation of three cars.
- The framework provides a solution to the challenges faced by the centralized machine learning approach in autonomous driving and enhances the users' privacy.

9.2 Future Scope

We are planning the following elements to be included in our future scope:

- Developing and testing model on new tracks
- Deploying the framework on real-time autonomous cars

REFERENCES

1. C. Nadiger, A. Kumar and S. Abdelhak, "Federated Reinforcement Learning for Fast Personalization," IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), 2019, pp. 123-127, doi: 10.1109/AIKE.2019.00031 (2019)
2. Loiacono, Daniele, Alessandro Prete, Pier Luca Lanzi and Luigi Cardamone. “Learning to overtake in TORCS using simple reinforcement learning.” IEEE Congress on Evolutionary Computation (2010)
3. L. Yi, "Lane Change of Vehicles Based on DQN," 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), pp. 593-597, doi: 10.1109/ISCTT51595.2020.00113 (2020)
4. Vitelli, M. Nayebi, A.: CARMA: A deep reinforcement learning approach to autonomous driving. Tech. rep. Stanford University, Tech. Rep., (2016)
5. T. Li, A. K. Sahu, A. Talwalkar and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," in IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50-60, May 2020, doi: 10.1109/MSP.2020.2975749
6. Qi, Jiaju, et al. "Federated reinforcement learning: Techniques, applications, and open challenges." Cornell University, arxiv.org/abs/2108.11887 (2021).
7. T.O.R.C.S. Manual installation and Robot tutorial: https://docplayer.net/28813194-T-o-r-c-s-manual-installation-and-robot-tutorial.html
8. TORCS website : http://torcs.sourceforge.net/index.php
9. Deep Reinforcement Learning framework for Autonomous Driving: https://arxiv.org/abs/1704.02532
10. Using Keras and Deep Deterministic Policy Gradient to play TORCS: https://yanpanlau.github.io/2016/10/11/Torcs-Keras.html

11. Torcs-Reinforcement-Learning: https://github.com/A-Raafat/Torcs---Reinforcement-Learning-using-Q-Learning/blob/master/README.md
12. An introduction to Policy Gradients with Cartpole and Doom- https://www.freecodecamp.org/news/an-introduction-to-policy-gradients-with-cartpole-and-doom-495b5ef2207f/#:~:text=Two%20types%20of%20policy,returns%20an%20action%20to%20take.&text=On%20the%20other%20hand%2C%20a,a%20probability%20distribution%20over%20actions
13. Using PyTorch and DDPG to play Torcs: https://github.com/jastfkjg/DDPG_Torcs_PyTorch
14. Deep Reinforcement Learning. Deep Deterministic Policy Gradient (DDPG) algorithm: https://markus-x-buchholz.medium.com/deep-reinforcement-learning-deep-deterministic-policy-gradient-ddpg-algoritm-5a823da91b43
15. TensorFlow Federated website: https://www.tensorflow.org/federated/federated_learning