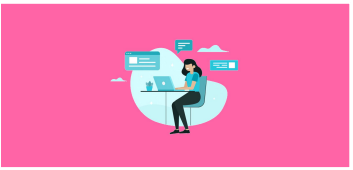
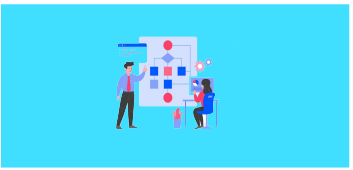







MKSSS's Cummins College of Engineering for Women, Pune

B. Tech Project ESE Review

Group - 16

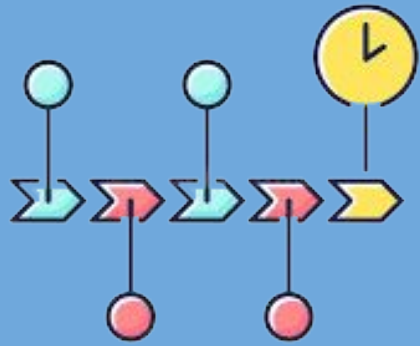
Presented By: Gayatri Walke Shuchi Talati Stuti Dhasmana Vaibhavi Kharapkar

Internal Guide: Dr. A.M. Naik External Guide: Mr. K Naveen Kumar

Sponsored By:

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Project Summary



NPTEL Course

Programming, Data Structures And Algorithms Using Python

Duration : 8 weeks

Marks: 50/50

Start Date : 24 Jan 2022

End Date : 18 Mar 2022

Week 1
Introduction to programming, algorithms and data structures, Python: variables, operations, control flow - assignments, condition-als, loops, functions

Week 2
Python: types, expressions, strings, lists, tuples
Python memory model, List operations
Inductive function definitions, Elementary inductive sorting: selection and insertion sort

Week 3
Basic algorithmic analysis: input size, asymptotic complexity, O() notation
Arrays vs lists, Merge sort, Quicksort, Stable sorting

Week 4
Dictionaries, Passing functions as arguments
Higher order functions on lists: map, lter, list comprehension

Week 5
Exception handling, Handling files
String processing

Week 6
Backtracking: N Queens, Scope in Python: local, global, nonlocal names
Nested functions, Data structures: stack, queue, Heaps

Week 7
Abstract data types, Binary search trees: find, insert, delete
Height-balanced binary search trees

Week 8
Efficient evaluation of recursive definitions: memoization
Dynamic programming: examples

Activity

Project Competitions

- **State Level Competition: Tech-Srujan** ***(29th April 2022)***, organized by MKSSS's Cummins College of Engineering for Women, Pune

- **Regional Level Project Competition** ***(2nd May 2022)***, organised by Director of Technical Education, Regional Office, Pune sponsored by IEEE communication Society and Signal Processing Society

Project Patent

- Project selected by Cummins College of Engineering for Women, Pune for patenting

Agenda

Agenda:

- Motivation and Purpose
- Background Study
- Project Specifications
- Module 1: The Racing Car Simulator
- Module 2: Reinforcement Learning Model
- Module 3: Federated Learning Model
- Module 4: Integration of Modules
- Technology and System Architecture
- ***Demonstration***
- Results & Conclusion
- References

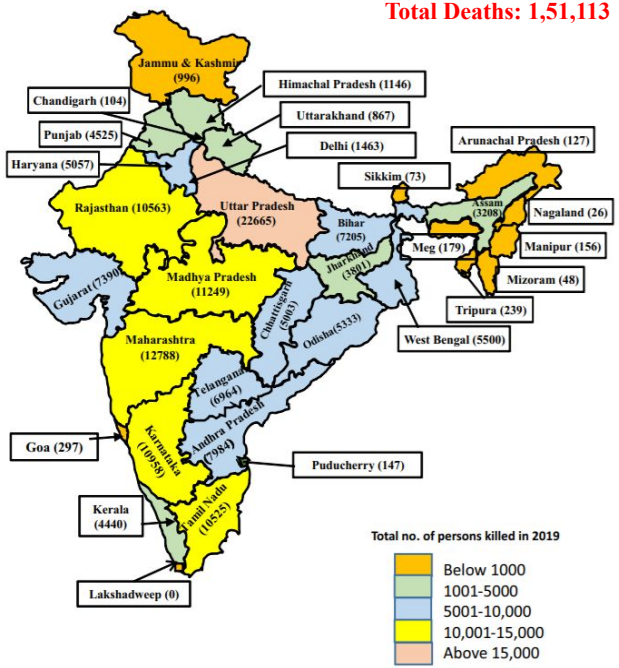
Motivation

Motivation

ROAD ACCIDENTS IN INDIA – 2019

Map 2.2 Persons Killed in Road Accidents in 2019 - State-Wise

Total Accidents: 4,49,002
Total Deaths: 1,51,113

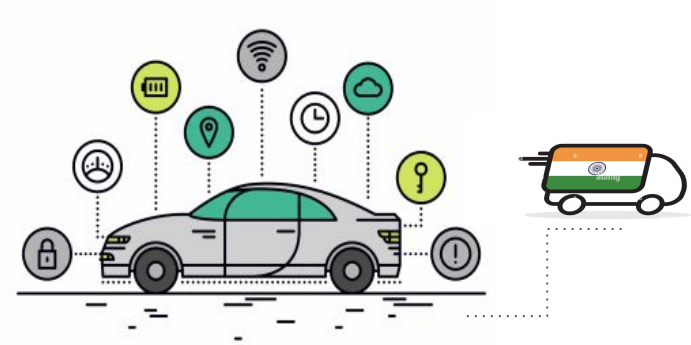


GOVERNMENT OF INDIA
MINISTRY OF ROAD TRANSPORT AND HIGHWAYS
TRANSPORT RESEARCH WING
NEW DELHI
www.morth.nic.in

Autonomous technology is not just a luxury but the need of the hour!

- With advancements in sensor technology and the maturity of algorithms, the tech will be better than **90% of drivers** out there in the next couple of years.

- India accounts for the highest number of road accidents globally and 400+ people die in road accidents every day.



Bringing autonomous vehicles to India

A Federated Reinforcement Learning framework for Autonomous Driving

Purpose

Ensure data privacy or data secrecy

Eliminate of single point of failure

Reduction in training and inference time

Scope/Out of Scope

Scope:

- Development of a **Federated Learning (FL)** framework to overcome statistical heterogeneity and ensure users' privacy
- Designing a **Reinforcement Learning (RL)** algorithm for accurate lane changing behaviour such as monitoring vehicle tracks
- Integration of individual models to generate a unified FRL framework for lane changing in autonomous vehicles
- Exploring TORC Simulator to deploy the developed **Federated Reinforcement Learning (FRL)** Model

Out of Scope:

- Focusing on multiple clients, considering only single vehicle case
- Developing or designing any new tracks, we are only using ones available in the TORCS

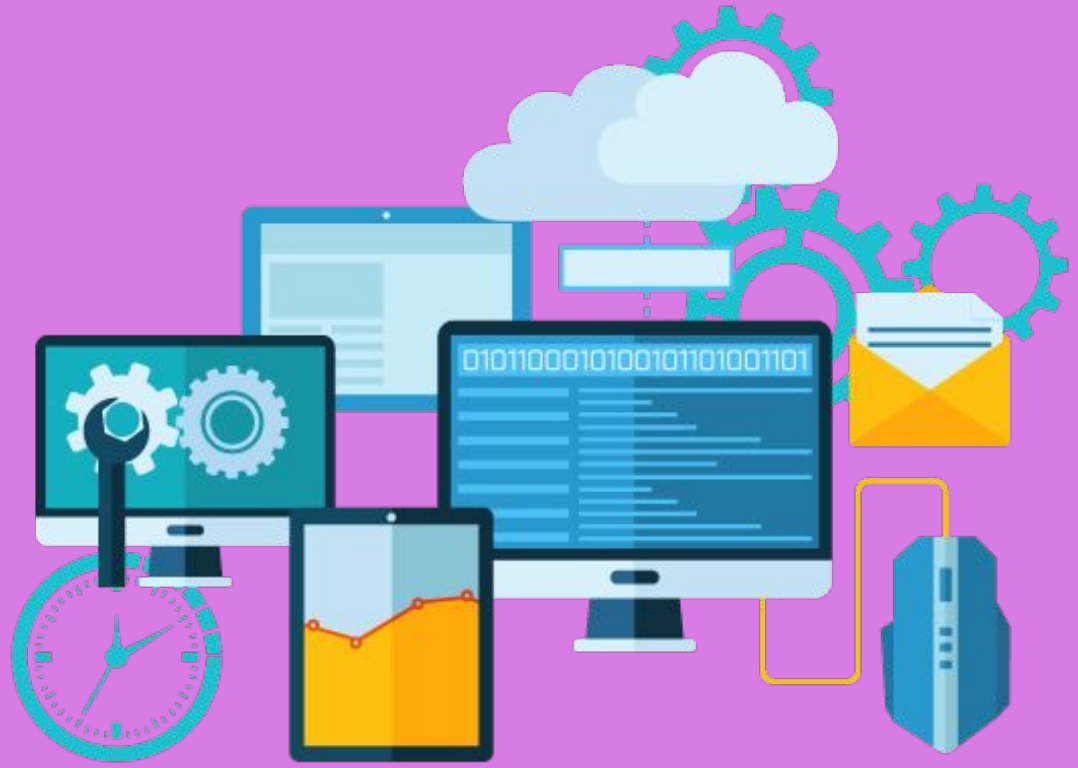
Literature Survey

S. No	Citation	Approach	Results/ Accuracy	Key Takeaway	Future Scope	Domain
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)	Agent is a DQN Model. The DQN training involved two neural networks namely Q and Target Q network	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%	Overall architecture for federated reinforcement learning (FRL), which includes the grouping policy, the learning policy, and the federation policy.	Extending the approach to areas where fast personalization in changing environment is needed	Federated learning, reinforcement learning; computer games, game personalization
2	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	Core challenges in FL, privacy preservation, survey of current work and future directions in FL	System heterogeneity might cause straggle the network. Privacy prevention comes at the cost of reduced system performance and efficiency.	Communication-efficiency, system heterogeneity, data heterogeneity, and privacy are the major challenges associated with Federated learning.	To find a robust methodology to overcome these major challenges in Federated learning	Federated Learning

S. No	Citation	Approach	Results/ Accuracy	Key Takeaway	Future Scope	Domain
3	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)	Reinforcement learning : Q learning with Behavior Analysis and Training (BAT) methodology	Results show that even simple Qlearning can produce competitive overtaking behaviors by 90.16% Success Rate	Reinforcement learning in TORCS - Reward function, Q table TORCS sensors, car state usage for programming	Integrating neural networks to implement Q learning to increase efficiency	Autonomous Driving , TORCS
4	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)	DQN, MAXMIN Q-Learning	Utilization of Q-learning to change lanes successfully validated on simulator Deep-trafic	DQN advantages over Q-learning for autonomous driving- reduce training time and storage space Different options for RL actions in simulator.	Multi step lane changing, real time application	DQN, Autonomous Driving

S. No	Citation	Approach	Results/ Accuracy	Key Takeaway	Future Scope	Domain
5	Vitelli, M. Nayebi, A.: CARMA: "**A deep reinforcement learning approach to autonomous driving**". Tech. rep. Stanford University, Tech. Rep., (2016)	Comparative analysis of different reinforcement learning methods to perform the task of autonomous car driving from raw sensory inputs	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.	Deep Q-Networks can be an effective means of controlling a vehicle directly from high-dimensional sensory inputs	A better alternative of defining their reward function that still maintains the careful balance between maximizing speed while guaranteeing car stability	Deep Q-network (DQN), Autonomous Driving
6	Z. Huang, J. Zhang, R. Tian and Y. Zhang, "**End-to-End Autonomous Driving Decision Based on Deep Reinforcement Learning**," 2019 5th International Conference on Control, Automation and Robotics (ICCAR), 2019, pp. 658-662, doi: 10.1109/ICCAR.2019.8813431.	Reinforcement learning: DDPG (Deep Deterministic Policy Gradient) with LRP (Layer-wise Relevance Propagation) algorithm.	The decision making control model directly outputs acceleration, braking, and steering behavior. DDPG agent determines the steering and acceleration through the front distance distribution, and determines the braking through the front distance and its own speed	Reinforcement learning in TORCS - DDPG network architecture and training strategy, Reward shaping, visualization DDPG neural network using LRP Algorithm	Visualize the critic network to explain which state and action affect the current Q value	DDPG, TORCS

Project Specifications

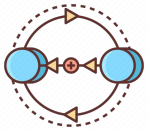


Modules

Module 1: TORC Simulator

Description: Deploying framework on TORCS through socket program to train and validate driver car agent with modern day methodology based on current car state and track information.

Module 2: Reinforcement Learning



Description: Train an autonomous agent on policy based approach in reinforcement learning with information captured in simulation environment.

FRL Framework

Module 3: Federated Learning

Description: To train the algorithm across multiple decentralised autonomous vehicles holding local data samples, without exchanging them with third party service. In contrast to centralised machine learning,

Module 4: Integration of Modules

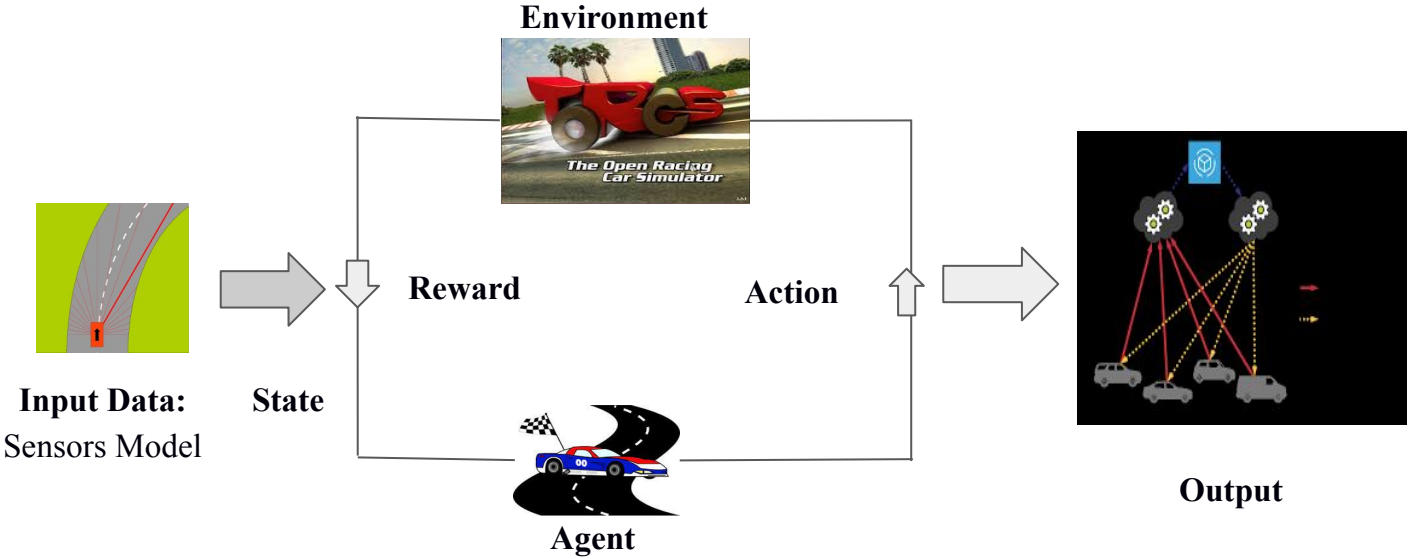
Description: Integration of individual models to generate a unified FRL framework for lane changing(monitoring vehicle track) in autonomous vehicle.

Module 1: The Open Racing Car Simulator (TORCS)

TORCS

The Open Racing Car Simulator

- Highly customizable, open source car racing simulator
- Provides a sophisticated physics engine, 3D graphics, and several diverse tracks and car models
- The car has access to all information of the environment
- The server acts as a proxy for the environment



TORCS

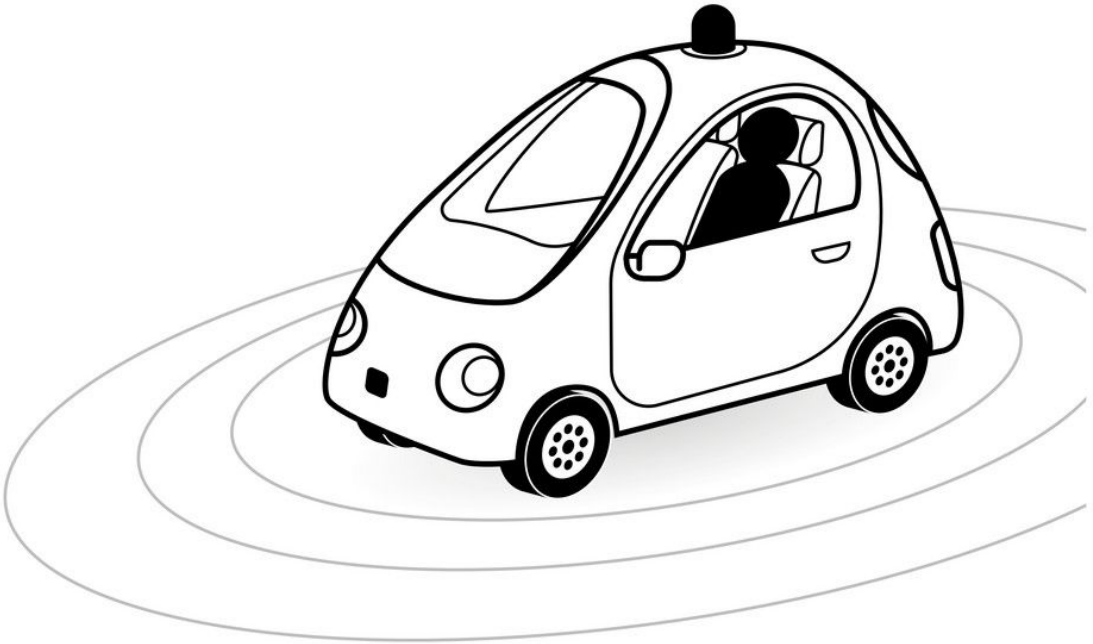
TORCS Sensors

- *Angle* 
- *Gear* 
- *speedX,Y*
- *distFromStartLine* 
- *distRaced*
- *trackPos* 
- *CurLapTime* 
- *lastLapTime*
- *wheelSpinVel* 
- *Fuel* 
- *Damage*
- *Rpm* 

TORCS

TORCS Control Actions

- ❑ *Acceleration*
- ❑ *Brake*
- ❑ *Gear*
- ❑ *Steering*
- ❑ *Meta*



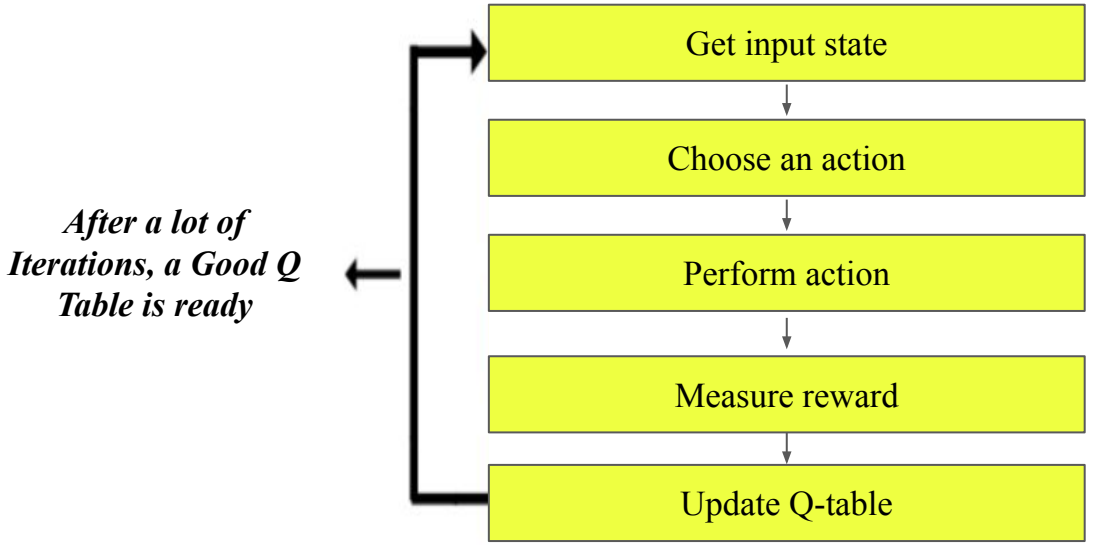
Module 2: Reinforcement Learning

Reinforcement Learning

Q-learning

- Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent.
- It learns the value function Q (S, a), which means how good to take action "**a**" at a particular state "**s**."

Algorithm



Reinforcement Learning

Input State[5]

- ***speedX:*** Speed of the car along the longitudinal axis of the car
- ***Angle:***Angle between the car direction and the direction of 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]

Actions[5]

- ***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
- ***Meta:*** This is meta-control command: 0 Do nothing, 1 ask server to restart the race

Rewards[5]

- 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 [-1,2]
- 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

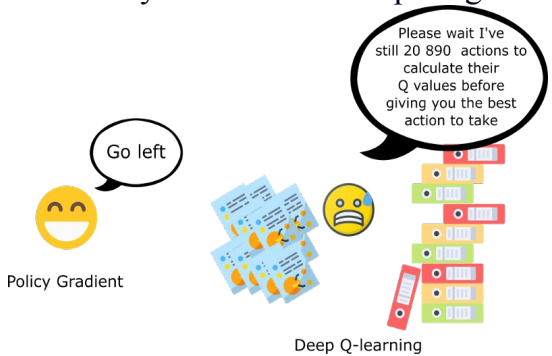
Q-Learning Demonstration

Reinforcement Learning

Why to use policy-based reinforcement learning methods over value based methods?

- **Policy-based methods have better convergence properties**
  - Value-based methods have a big oscillation while training
  - In policy gradient, we follow the gradient to find the best parameters

- **Policy gradients are more effective in high dimensional action spaces, or when using continuous actions**
  - In case of a self driving car, at each state you can have an infinite choice of actions (turning the wheel at 15°, 17.2°, 19,4°…). Therefore, we need to output a Q-value for each possible action
  - In policy-based methods, you just adjust the parameters directly rather than computing the maximum expected future reward at every step



Reinforcement Learning

Deep Deterministic Policy Gradient (DDPG)[8]

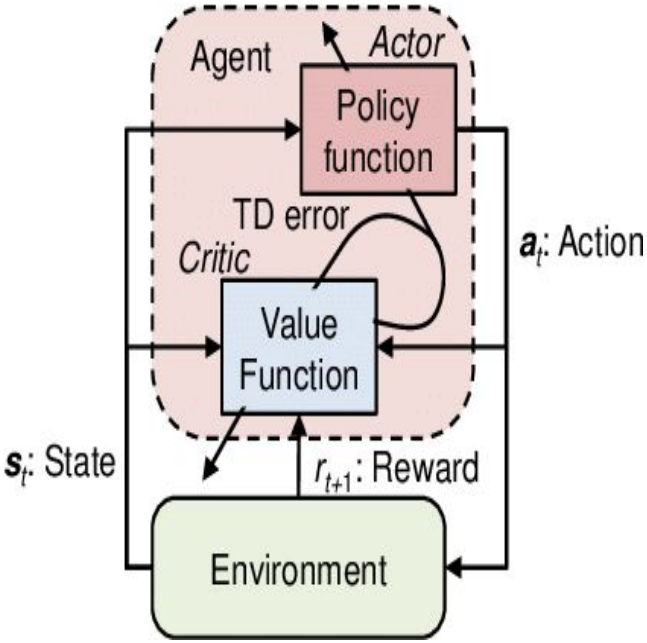
1. The **Actor network** takes input state s and results in the action a.

2. The **Critic network** 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 the Critic network weight with the gradients calculated from the **Temporal Difference** (TD) error.

5. Selected action in a state, s, receive a reward, r and move to a new state, s'.

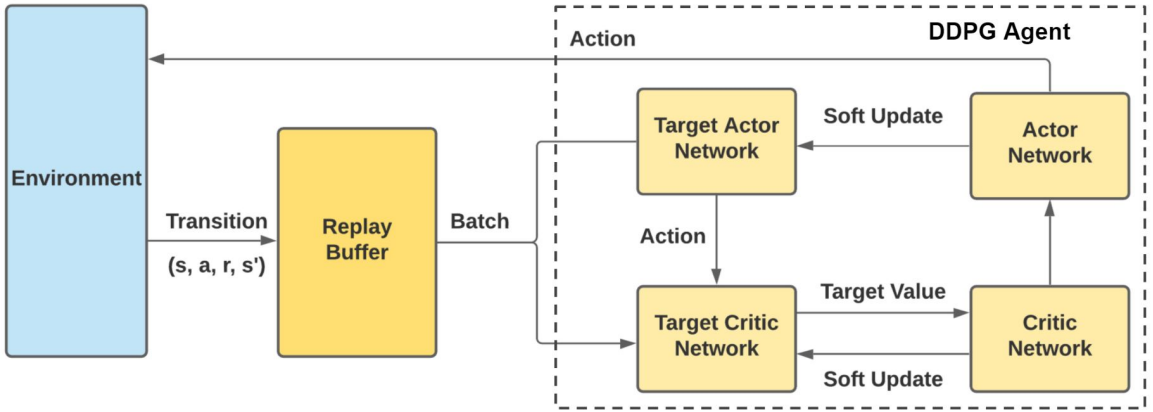


Reinforcement Learning

Deep Deterministic Policy Gradient (DDPG)[8]

6. Store this transition information in an experience **replay buffer**.
7. 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.
8. Update our **policy network** weights using a policy gradient.
9. Update the weights of Actor and Critic network in the **target network**.

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

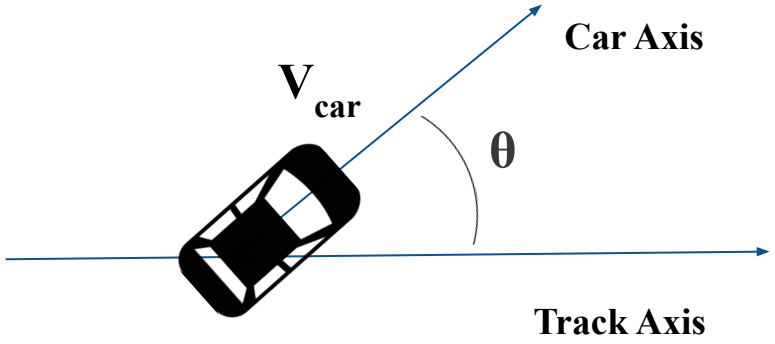


Reinforcement Learning

Reward Function

The reward function is below:

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



Termination Judgement

Episode terminates if

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

Collision Detection

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

Reinforcement Learning

How do we do exploration in the continuous action space?

Ornstein-Uhlenbeck (OU) Process

It is a stochastic process which has mean-reverting properties.

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t$$

- θ means the how "fast" the variable reverts towards to the mean
- μ represents the equilibrium or mean value
- σ is the degree of volatility of the process

The following table shows the values:

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

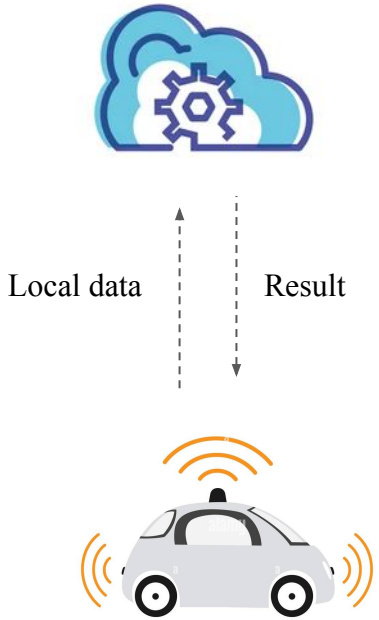
Deep Deterministic Policy Gradient (DDPG) Demonstration

0.98355 0.98907 0.97655505
0.981755 0.987265 0.974775
0.981755 0.987265 0.974775
0.97995 0.98544997 0.97297996
0.97995 0.98544997 0.97297996
0.978135 0.98363 0.97118497
0.978135 0.98363 0.97118497
0.97630996 0.98179495 0.969375
0.97630996 0.98179495 0.969375
0.974475 0.97995496 0.967555
0.974475 0.97995496 0.967555
0.972635 0.978105 0.96572995
0.972635 0.978105 0.96572995
0.97079 0.976245 0.963895
0.97079 0.976245 0.963895
0.96892 0.97436994 0.962035
0.96892 0.97436994 0.962035
0.96705496 0.9725 0.960185
0.96705496 0.9725 0.960185
0.965185 0.97062 0.95833
0.965185 0.97062 0.95833
0.96331 0.968735 0.956475
0.96331 0.968735 0.956475
0.961425 0.96684504 0.95460504
0.961425 0.96684504 0.95460504
0.95954 0.964945 0.952735
0.95954 0.964945 0.952735
0.95764506 0.96304494 0.950855
0.95764506 0.96304494 0.950855
0.95575 0.96114 0.94897497
0.95575 0.96114 0.94897497
0.95385003 0.959225 0.94708997
0.95385003 0.959225 0.94708997
0.95194 0.95731 0.945195
0.95194 0.95731 0.945195
0.95002997 0.95539004 0.94330496
0.95002997 0.95539004 0.94330496
0.94811994 0.95347 0.94140506
0.94811994 0.95347 0.94140506
0.9462 0.95154 0.939505
0.9462 0.95154 0.939505
0.94428 0.94961 0.9376
0.94428 0.94961 0.9376
0.942355 0.94768 0.93569
0.942355 0.94768 0.93569
0.94043 0.945745 0.93377995
0.94043 0.945745 0.93377995
0.938505 0.943805 0.93187
0.938505 0.943805 0.93187
0.936575 0.941865 0.929955
0.936575 0.941865 0.929955
0.934645 0.939925 0.92804

Module 3: Federated Learning

Federated Learning

Machine Learning Conventional Approach

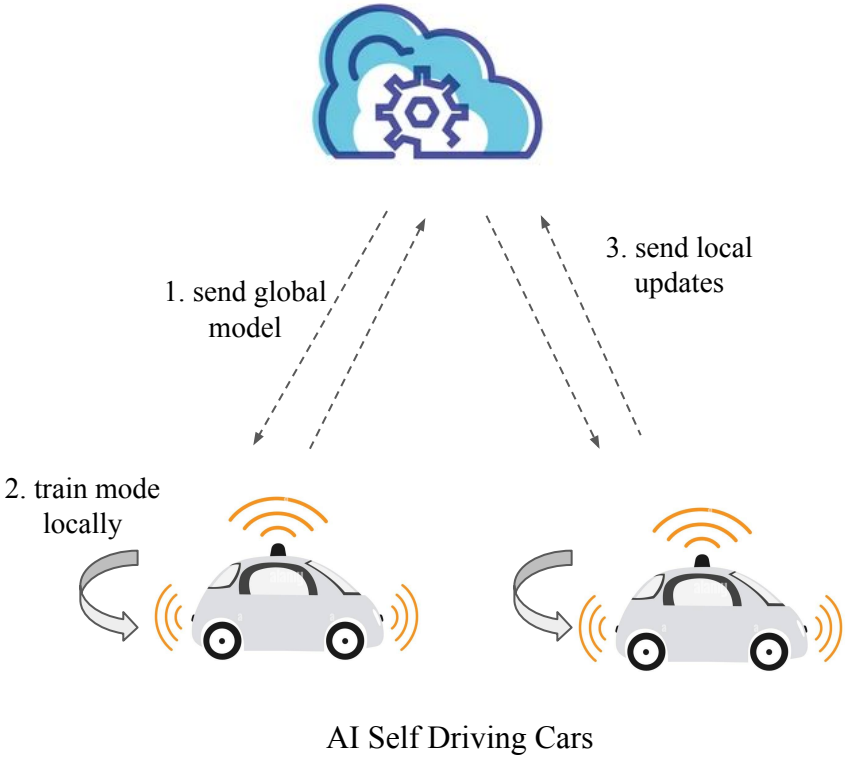


- Machine Learning(ML) occurs on cloud
- Global dataset is in cloud
- Collects together all data

- high latency
- compromises user’s privacy

Federated Learning

Machine Learning Federated Approach



AI Self Driving Cars

- Machine Learning occurs in the cloud and at edge devices(in AI self-driving cars)

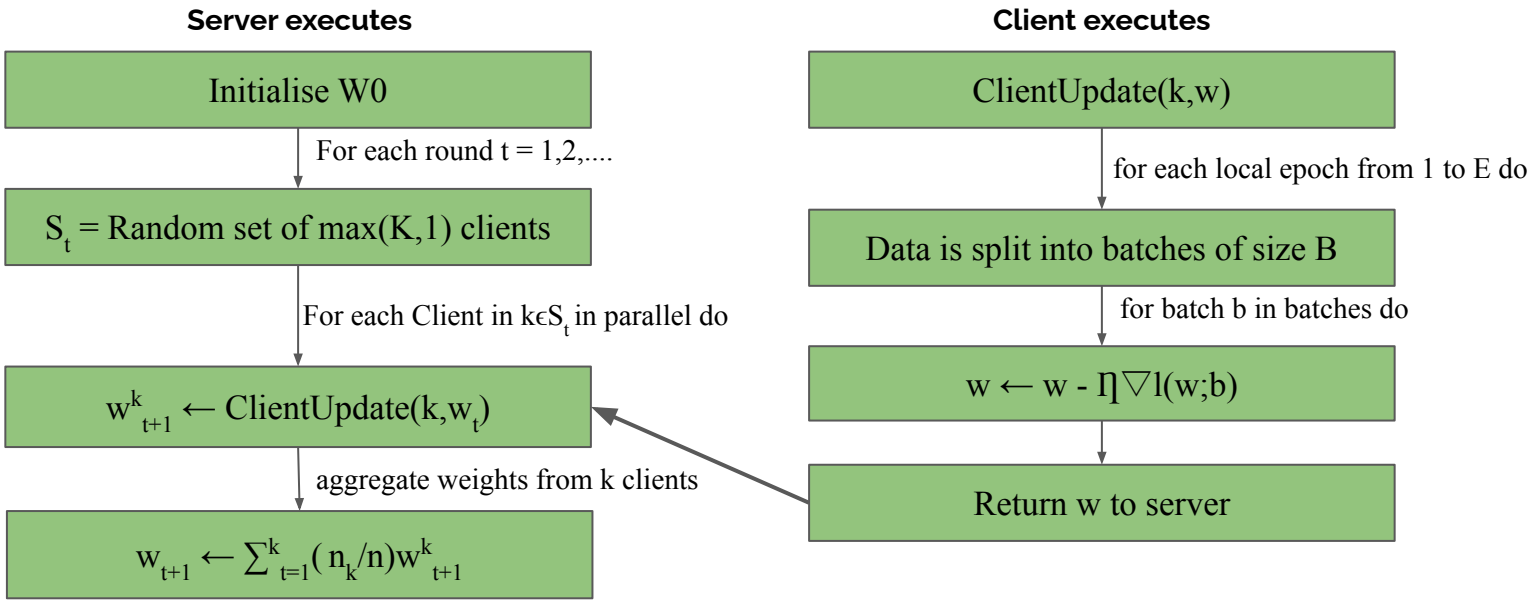
Benefits for Federated Learning:

- Decentralized and collaborative learning
- Ensures user privacy
- Lower Latency
- Reduces hardware cost

- Local data
- ML model = Localized Learning

Federated Learning

FedAvg - Algorithm for Federated Learning



Aggregation Function

$$\min_w F(w), \text{ where } F(w) := \sum_{k=1}^{m} p_k F_k(w)$$

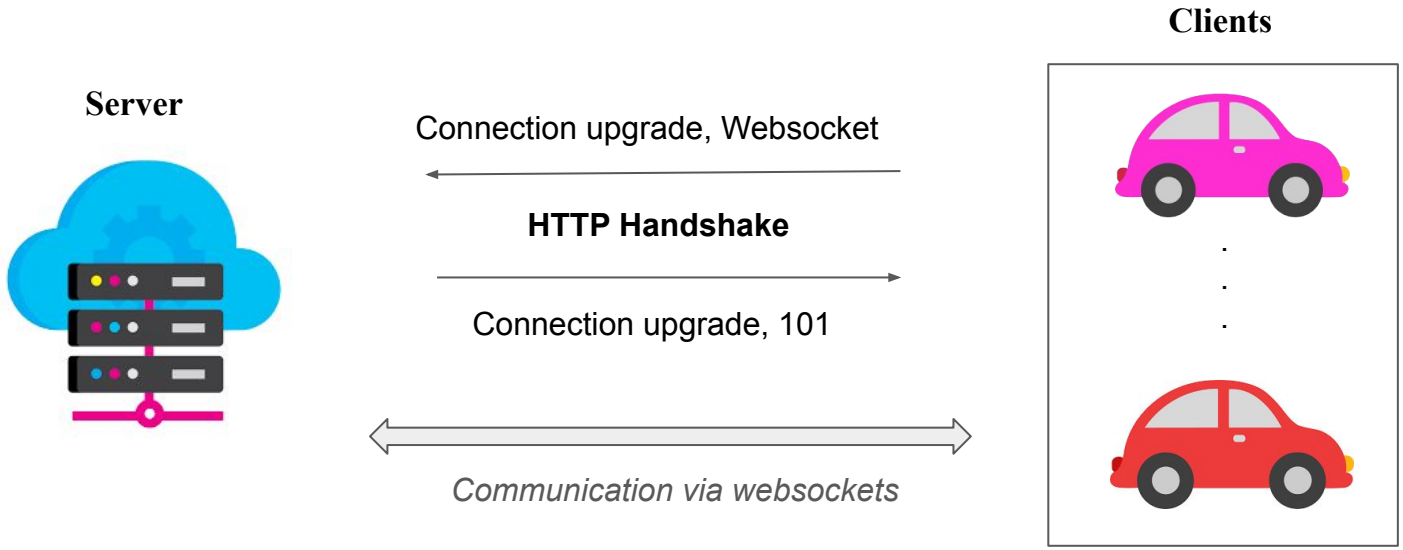
Here m is the total number of devices, F_k is the local objective function for the k^{th} device, and p_k specifies the relative impact of each device with $p_k >= 0$.

Module 4: Integration of Modules

Integration of Modules

WebSocket

- Persistent connection between a client and server
- Provides a bidirectional, full-duplex communications channel
- Operates over HTTP through a single TCP/IP socket connection



System Architecture & Technology

System Architecture

$$\sum_{k=1}^{K} \frac{n_k}{n}$$

Model updates are aggregated & updated global model is sent to edge devices

Websocket

TORCS Server

Sensory Input

Output of Actuators

(UDP)

RL Agent

Client 1

Client 2

Client 3

Client k

Perform local training

Technology

Operating System:
Windows

Simulator Environment:
TORC 3D Simulator

Programming Language:
Python

IDE:
Atom, PyCharm

Server:
Linode

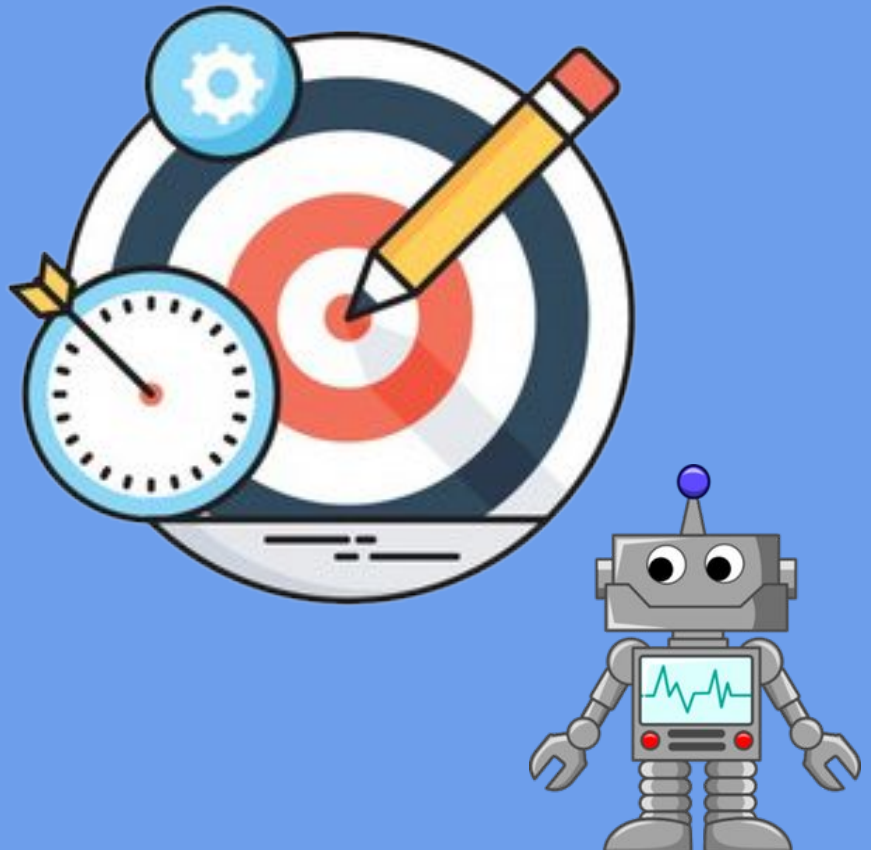
Softwares:
Git, Putty

Libraries:

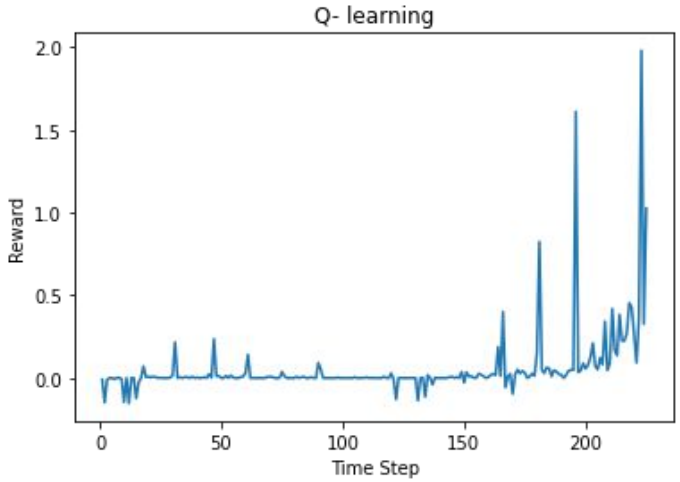
- ***pytorch***: provides a collection of workflows to develop and train RL models
- ***matplotlib***: providing the measurements and visualizations needed during the machine learning workflow
- ***Socket***: connects the server host given a specific port and client.
- ***Pandas***: structure data (for importing and analyzing data)
- ***gymTorcs:*** provides connection between TORCS server and code

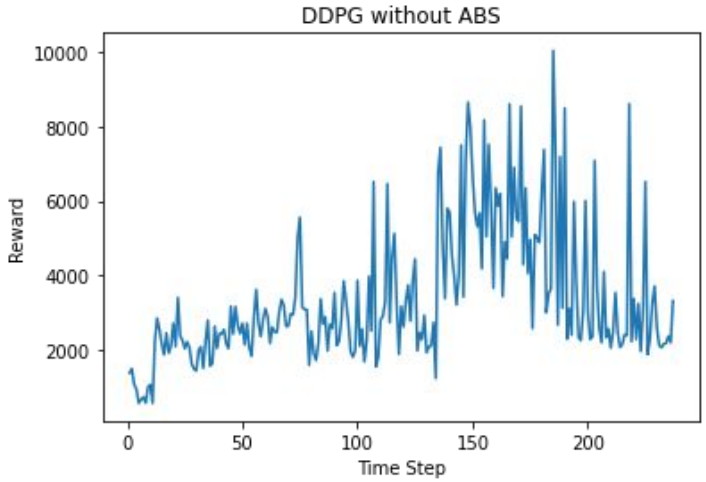
Demonstration

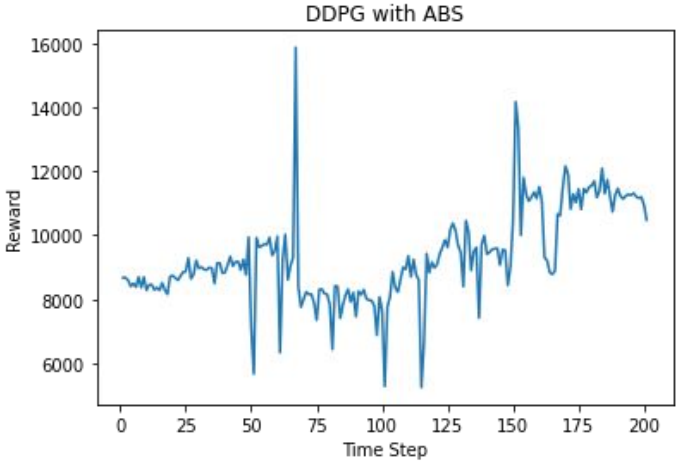
Results & Conclusion

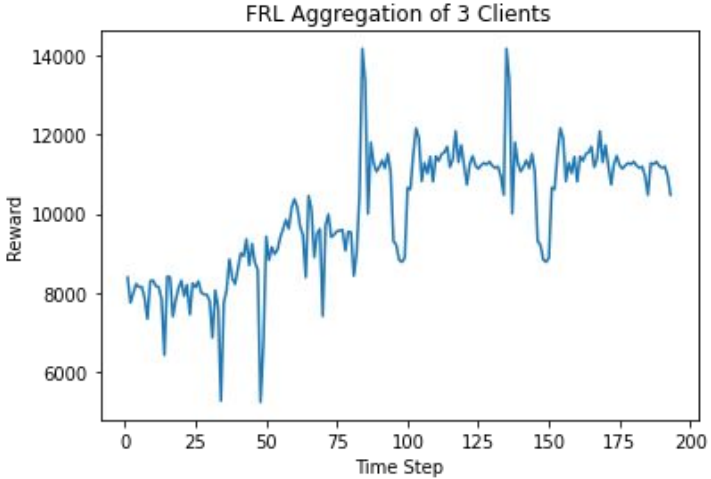


Results


Q- learning
Reward
Time Step
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200


DDPG without ABS
Reward
Time Step
10000
8000
6000
4000
2000
0
50
100
150
200


DDPG with ABS
Reward
Time Step
16000
14000
12000
10000
8000
6000
0
25
50
75
100
125
150
175
200

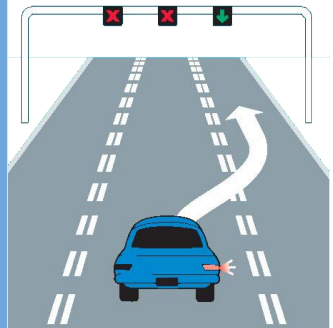

FRL Aggregation of 3 Clients
Reward
Time Step
14000
12000
10000
8000
6000
0
25
50
75
100
125
150
175
200

Conclusion

DDPG handles continuous action spaces efficiently without losing adequate exploration as compared to value based approaches

FRL-DDPG performs better than single execution of single RL agents

Results indicate that with FRL framework can accelerate the training speed and improve the performance of the federation of multiple cars



"Lane changing is one out the many aspects of self-driven vehicles, this project aims to elevate its level to all these aspects and bringing a positive change in the world of Autonomous Driving"

Project Timeline

SEMESTER - I

PROJECT REVIEW- I

Problem Statement, Motivation, Scope, Literature Survey, Project Specifications

24th December 2021

Demonstration of Q-Learning Algorithm, SRS, System Architecture, Agile Methodology

PROJECT REVIEW- II

24th January 2022

ESE REVIEW- I

Summary of Sem 1 Review 1 + Review 2 to External Examiner

4th February 2022

SEMESTER - II

Demonstration of DDPG Algorithm, FL Algorithm

PROJECT REVIEW- I

16th March 2022

PROJECT REVIEW- II

Integration of modules and Demonstration of FRL Framework Research Paper

4th May 2022

ESE REVIEW- II

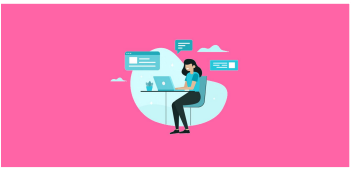
Project Completion

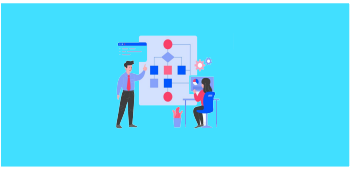
2nd June 2022

References

1. *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*
2. *TORCS website : http://torcs.sourceforge.net/index.php*
3. *Deep Reinforcement Learning framework for Autonomous Driving : https://arxiv.org/abs/1704.02532*
4. *Using Keras and Deep Deterministic Policy Gradient to play TORCS : https://yanpanlau.github.io/2016/10/11/Torcs-Keras.html*
5. *Torcs - Reinforcement-Learning : https://github.com/A-Raafat/Torcs---Reinforcement-Learning-using-Q-Learning/blob/master/README.md*
6. 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
7. *Using PyTorch and DDPG to play Torcs : https://github.com/jastfkjg/DDPG_Torcs_PyTorch*
8. *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*
9. *Federated Learning Github: https://github.com/kiddyboots216/FedRL/*
10. *Qi, Jiaju, et al. "Federated reinforcement learning: Techniques, applications, and open challenges." Cornell University, arxiv.org/abs/2108.11887 (2021)*








Thank You