<u>**Results**</u>

1) Q-Learning Algorithm:

1.1 <u>For epsilon = 0.9, discount-factor = 0.9, learning-rate = 0.1:</u>

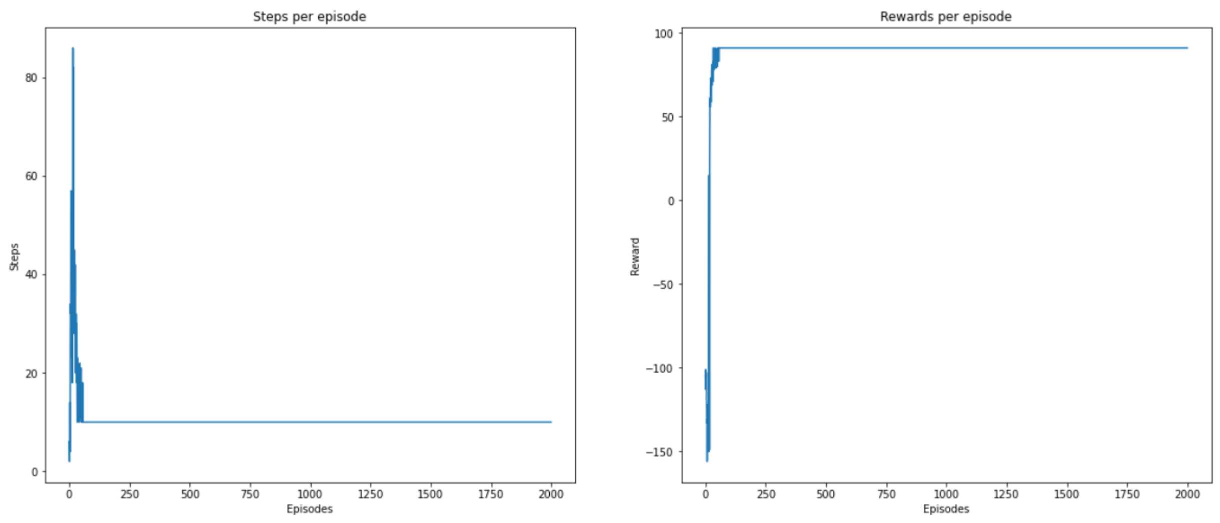


Fig 3- Comparison graphs for Steps vs Episodes and Rewards vs Episodes for epsilon=0.9 in Q-Learning

- In the first graph we can see a steep decrease in the graph where we visualize the Steps per Episode showing the number of steps performed in each episode decreasing steeply and going constant after certain steps.
- The second graph showing the Rewards per Episode we can see the rewards earned by the learning agent showing a sudden increase after a few episodes only.

1.2 <u>For epsilon = 1, discount-factor = 0.9, learning-rate = 0.1:</u>

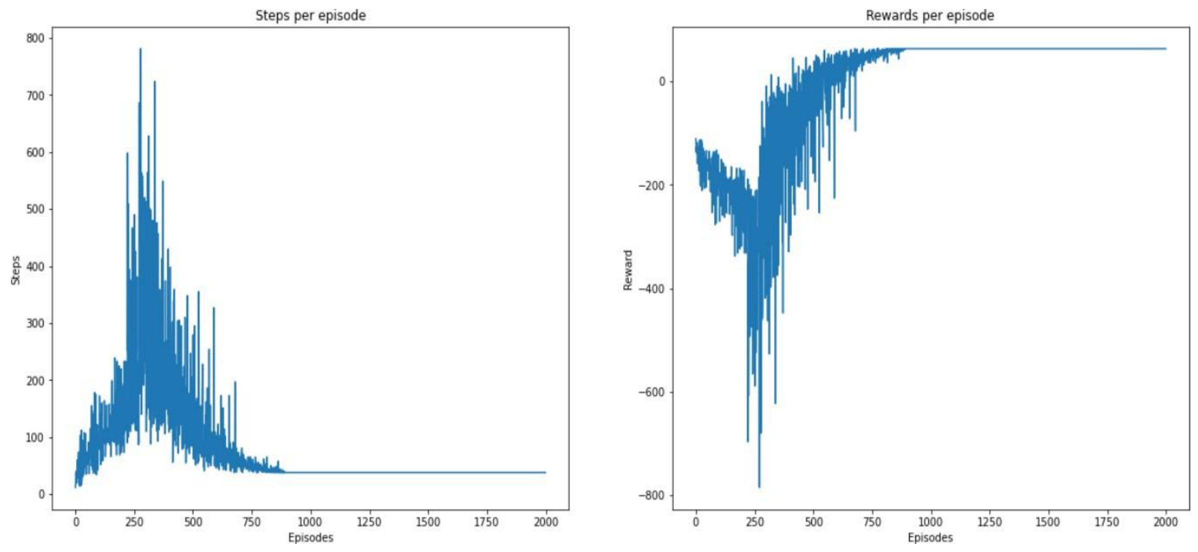


Fig 4- Comparison graphs for Steps vs Episodes and Rewards vs Episodes for epsilon=1.0 in Q-Learning

- The first graph for Steps per Episodes shows that with the number of episodes increasing the steps increase steadily and then decrease in the same manner and eventually become constant signifying that the learning agent has found a path to the end state.

- In the second graph of Rewards per Episodes as well we can see a steady decrease in the rewards as the episodes increases and then they increase steadily till the agent reaches the end state.

2) SARSA Algorithm:

2.1 For epsilon = 0.9, discount-factor = 0.9, learning-rate = 0.1

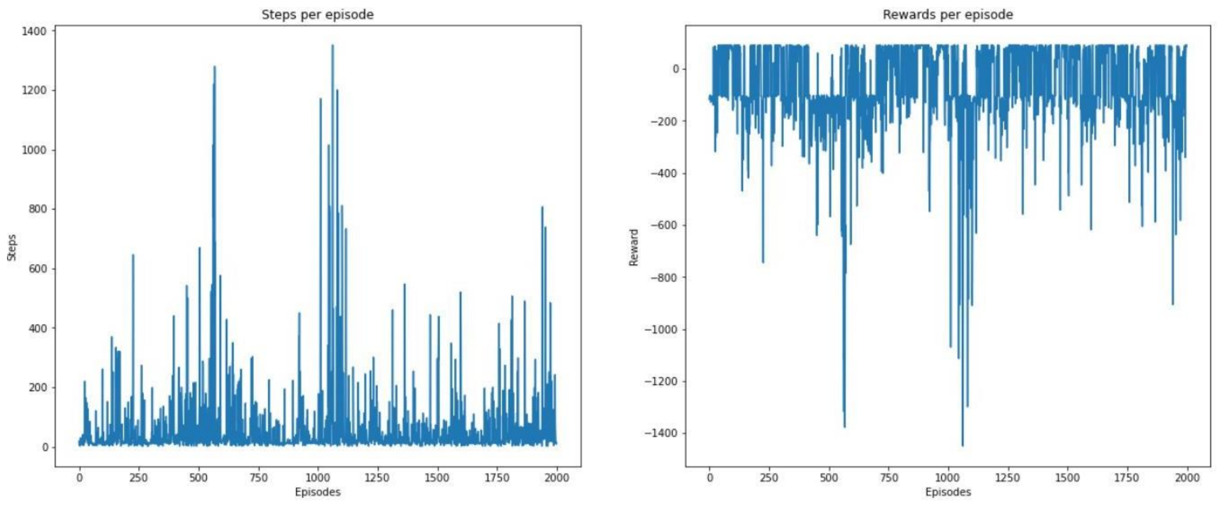


Fig 5- Comparison graphs for Steps vs Episodes and Rewards vs Episodes for epsilon=0.9 in Sarsa

- While performing the experiment for 0.9 the obstacle percentage had to be reduced to approximately 15% as the algorithm was not able to devise a path when we had set some exploration rate in the hyper-parameter epsilon. This percentage was set to be around 25-30% in Q-learning.
- In the first graph showing Steps per Episode, we can see clear variations in the steps as the number of episodes go on increasing.
- The similar fashion is observable in the second graph where we see the rewards earned with each episode showing variations till the agent reaches the end goal.

2.2 For epsilon = 1, discount-factor = 0.9, learning-rate = 0.1

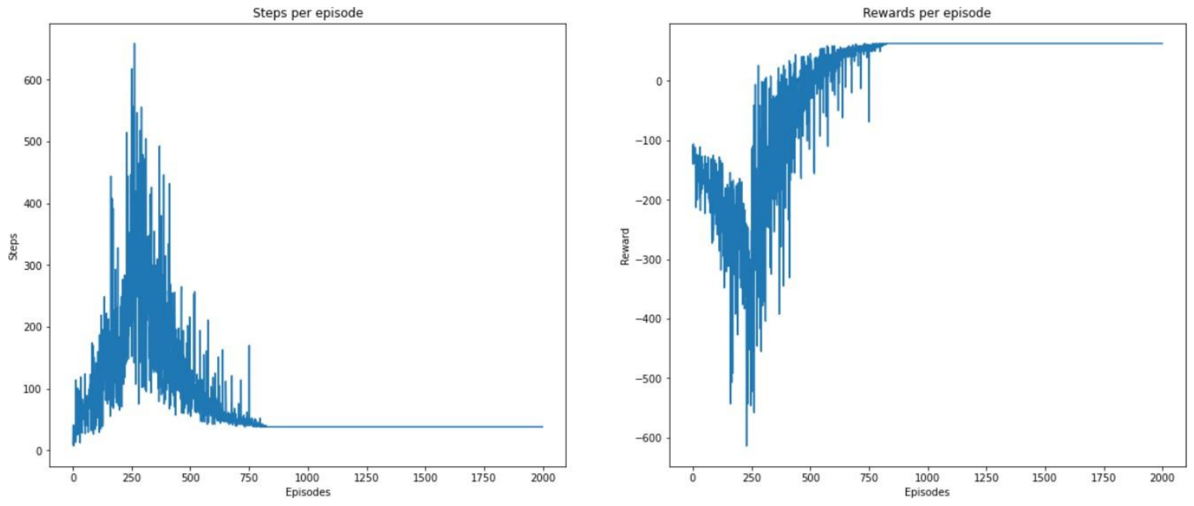
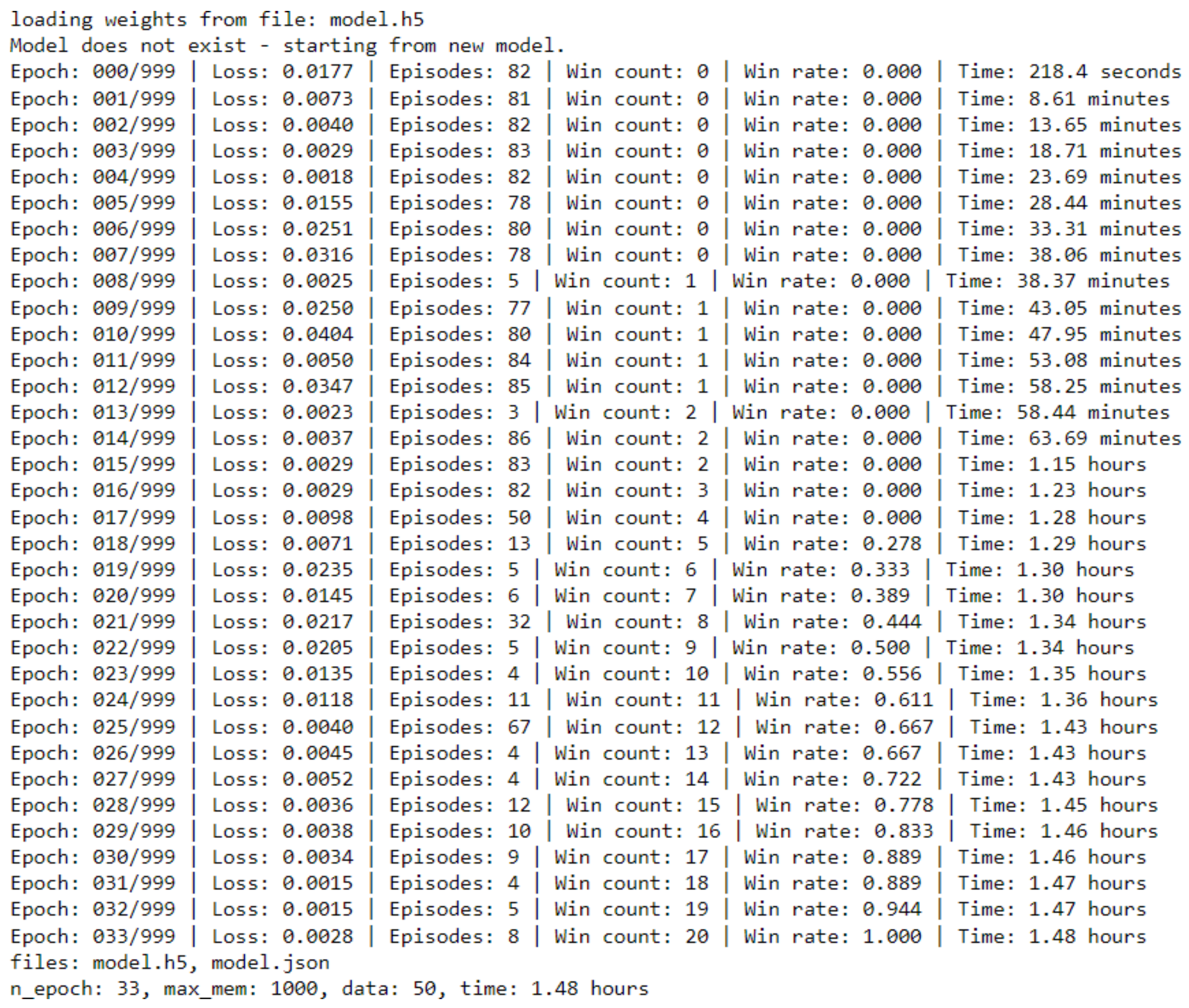


Fig 6- Comparison graphs for Steps vs Episodes and Rewards vs Episodes for epsilon=1.0 in Sarsa

- The first graph for Steps per Episodes shows that with the number of episodes increasing the steps increase steadily and then decrease in the same manner and eventually become constant signifying that the learning agent has found a path to the end state.
- In the second graph of Rewards per Episodes as well we can see a steady decrease in the rewards as the episodes increases and then they increase steadily till the agent reaches the end state.

3) Deep-Q Learning:



```
loading weights from file: model.h5
Model does not exist - starting from new model.
Epoch: 000/999 | Loss: 0.0177 | Episodes: 82 | Win count: 0 | Win rate: 0.000 | Time: 218.4 seconds
Epoch: 001/999 | Loss: 0.0073 | Episodes: 81 | Win count: 0 | Win rate: 0.000 | Time: 8.61 minutes
Epoch: 002/999 | Loss: 0.0040 | Episodes: 82 | Win count: 0 | Win rate: 0.000 | Time: 13.65 minutes
Epoch: 003/999 | Loss: 0.0029 | Episodes: 83 | Win count: 0 | Win rate: 0.000 | Time: 18.71 minutes
Epoch: 004/999 | Loss: 0.0018 | Episodes: 82 | Win count: 0 | Win rate: 0.000 | Time: 23.69 minutes
Epoch: 005/999 | Loss: 0.0155 | Episodes: 78 | Win count: 0 | Win rate: 0.000 | Time: 28.44 minutes
Epoch: 006/999 | Loss: 0.0251 | Episodes: 80 | Win count: 0 | Win rate: 0.000 | Time: 33.31 minutes
Epoch: 007/999 | Loss: 0.0316 | Episodes: 78 | Win count: 0 | Win rate: 0.000 | Time: 38.06 minutes
Epoch: 008/999 | Loss: 0.0025 | Episodes: 5 | Win count: 1 | Win rate: 0.000 | Time: 38.37 minutes
Epoch: 009/999 | Loss: 0.0250 | Episodes: 77 | Win count: 1 | Win rate: 0.000 | Time: 43.05 minutes
Epoch: 010/999 | Loss: 0.0404 | Episodes: 80 | Win count: 1 | Win rate: 0.000 | Time: 47.95 minutes
Epoch: 011/999 | Loss: 0.0050 | Episodes: 84 | Win count: 1 | Win rate: 0.000 | Time: 53.08 minutes
Epoch: 012/999 | Loss: 0.0347 | Episodes: 85 | Win count: 1 | Win rate: 0.000 | Time: 58.25 minutes
Epoch: 013/999 | Loss: 0.0023 | Episodes: 3 | Win count: 2 | Win rate: 0.000 | Time: 58.44 minutes
Epoch: 014/999 | Loss: 0.0037 | Episodes: 86 | Win count: 2 | Win rate: 0.000 | Time: 63.69 minutes
Epoch: 015/999 | Loss: 0.0029 | Episodes: 83 | Win count: 2 | Win rate: 0.000 | Time: 1.15 hours
Epoch: 016/999 | Loss: 0.0029 | Episodes: 82 | Win count: 3 | Win rate: 0.000 | Time: 1.23 hours
Epoch: 017/999 | Loss: 0.0098 | Episodes: 50 | Win count: 4 | Win rate: 0.000 | Time: 1.28 hours
Epoch: 018/999 | Loss: 0.0071 | Episodes: 13 | Win count: 5 | Win rate: 0.278 | Time: 1.29 hours
Epoch: 019/999 | Loss: 0.0235 | Episodes: 5 | Win count: 6 | Win rate: 0.333 | Time: 1.30 hours
Epoch: 020/999 | Loss: 0.0145 | Episodes: 6 | Win count: 7 | Win rate: 0.389 | Time: 1.30 hours
Epoch: 021/999 | Loss: 0.0217 | Episodes: 32 | Win count: 8 | Win rate: 0.444 | Time: 1.34 hours
Epoch: 022/999 | Loss: 0.0205 | Episodes: 5 | Win count: 9 | Win rate: 0.500 | Time: 1.34 hours
Epoch: 023/999 | Loss: 0.0135 | Episodes: 4 | Win count: 10 | Win rate: 0.556 | Time: 1.35 hours
Epoch: 024/999 | Loss: 0.0118 | Episodes: 11 | Win count: 11 | Win rate: 0.611 | Time: 1.36 hours
Epoch: 025/999 | Loss: 0.0040 | Episodes: 67 | Win count: 12 | Win rate: 0.667 | Time: 1.43 hours
Epoch: 026/999 | Loss: 0.0045 | Episodes: 4 | Win count: 13 | Win rate: 0.667 | Time: 1.43 hours
Epoch: 027/999 | Loss: 0.0052 | Episodes: 4 | Win count: 14 | Win rate: 0.722 | Time: 1.43 hours
Epoch: 028/999 | Loss: 0.0036 | Episodes: 12 | Win count: 15 | Win rate: 0.778 | Time: 1.45 hours
Epoch: 029/999 | Loss: 0.0038 | Episodes: 10 | Win count: 16 | Win rate: 0.833 | Time: 1.46 hours
Epoch: 030/999 | Loss: 0.0034 | Episodes: 9 | Win count: 17 | Win rate: 0.889 | Time: 1.46 hours
Epoch: 031/999 | Loss: 0.0015 | Episodes: 4 | Win count: 18 | Win rate: 0.889 | Time: 1.47 hours
Epoch: 032/999 | Loss: 0.0015 | Episodes: 5 | Win count: 19 | Win rate: 0.944 | Time: 1.47 hours
Epoch: 033/999 | Loss: 0.0028 | Episodes: 8 | Win count: 20 | Win rate: 1.000 | Time: 1.48 hours
files: model.h5, model.json
n_epoch: 33, max_mem: 1000, data: 50, time: 1.48 hours
```

Conclusion

- The **Q-Learning** algorithm is an off-policy reinforcement learning algorithm where the agent uses a greedy approach by selecting the maximum rewards in the existing Q-Table. The experiments performed using Q-Learning show that it is a very aggressive learning algorithm that doesn't penalize the mistakes much as compared to the Sarsa algorithm as well as on average it takes fewer steps compared to the Sarsa algorithm. We can use this learning algorithm at places where we need quick and optimal results but there is always an uncertainty of failure, as much risk is involved. Game Playing is a good application where we can apply Q-Learning like in playing chess.
- The **SARSA** algorithm is an on-policy reinforcement learning algorithm and is a much more conservative learner as compared to the Q-Learning algorithm. The experiments performed show that it penalizes more on the mistakes as compared to the Q-Learning algorithm and always tries to avoid the obstacle-laid path as much as possible. We can

use this algorithm at places where we cannot bear many failures and we need to penalize each mistake heavily to find a safe path. We can apply this algorithm to applications that are based on the Cliff-Walking concept.

- The **Deep-Q Learning** approach is used when the number of input states in a certain application cannot be handled by the use of a Q-Table. This approach finds a good path on the input and can handle large state spaces easily but takes a lot of time to train on a given input state space which could be tackled by making certain optimizations in the algorithm and also by using a good GPU to train the network on the training data.

REFERENCES

[1] X. Wang, L. Jin and H. Wei, "The Shortest Path Planning Based on Reinforcement Learning", Journal of Physics: Conference Series, vol. 1584, no. 1, p. 012006, 2020.

[2] A. Panov, K. Yakovlev, and R. Suvorov, "Grid Path Planning with Deep Reinforcement Learning: Preliminary Results", Procedia Computer Science, vol. 123, pp. 347-353, 2018.

[3] C. Chen, X. Chen, F. Ma, X. Zeng, and J. Wang, "A knowledge-free path planning approach for smart ships based on reinforcement learning", Ocean Engineering, vol. 189, p. 106299, 2019.

[4] Luo, Wei & Tang, Qirong & Fu, Changhong & Eberhard, Peter. (2018). Deep-Sarsa Based Multi-UAV Path Planning and Obstacle Avoidance in a Dynamic Environment. 10.1007/978-3-319-93818-9_10.

[5] C. You, J. Lu, D. Filev and P. Tsiotras, "Highway Traffic Modeling and Decision Making for Autonomous Vehicle Using Reinforcement Learning," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 1227-1232, DOI: 10.1109/IVS.2018.8500675.

[6] D. Xu, Y. Fang, Z. Zhang, and Y. Meng, "Path Planning Method Combining Depth Learning and Sarsa Algorithm," 2017 10th International Symposium on Computational Intelligence and Design (ISCID), 2017, pp. 77-82, DOI: 10.1109/ISCID.2017.145.

[7] Guan, Meng & Yang, Fu & Jiao, Ji & Chen, Xin. (2021). Research on path planning of mobile robot based on improved Deep Q Network. Journal of Physics: Conference Series. 1820. 012024. 10.1088/1742-6596/1820/1/012024.

[8] P. Mohan, P. Narayan, L. Sharma, T. Jambhale and S. Koul, "Iterative SARSA: The Modified SARSA Algorithm for Finding the Optimal Path", International Journal of Recent Technology and Engineering, vol. 8, no. 6, pp. 4333-4338, 2020.