AUTHOR

Ata Ertuğ Çil
A.E.Cil@student.tudelft.nl

Effects of Partial Observability Solver Methods on Training and Final Policies in Autonomous Driver RL

SUPERVISORS

Moritz Zanger
Matthijs Spaan

01. Introduction

Autonomous driving is a complex problem that can be solved using artificial intelligence. However as the environment is not fully observable and changing constantly, many different methods are investigated to be able to make agent comprehends the state fully. In this study, we focused on methods that solve partial observability.

02. Research question

How do different methods for dealing with partial observability in the environment influence training and the robustness of final policies under various testing conditions?

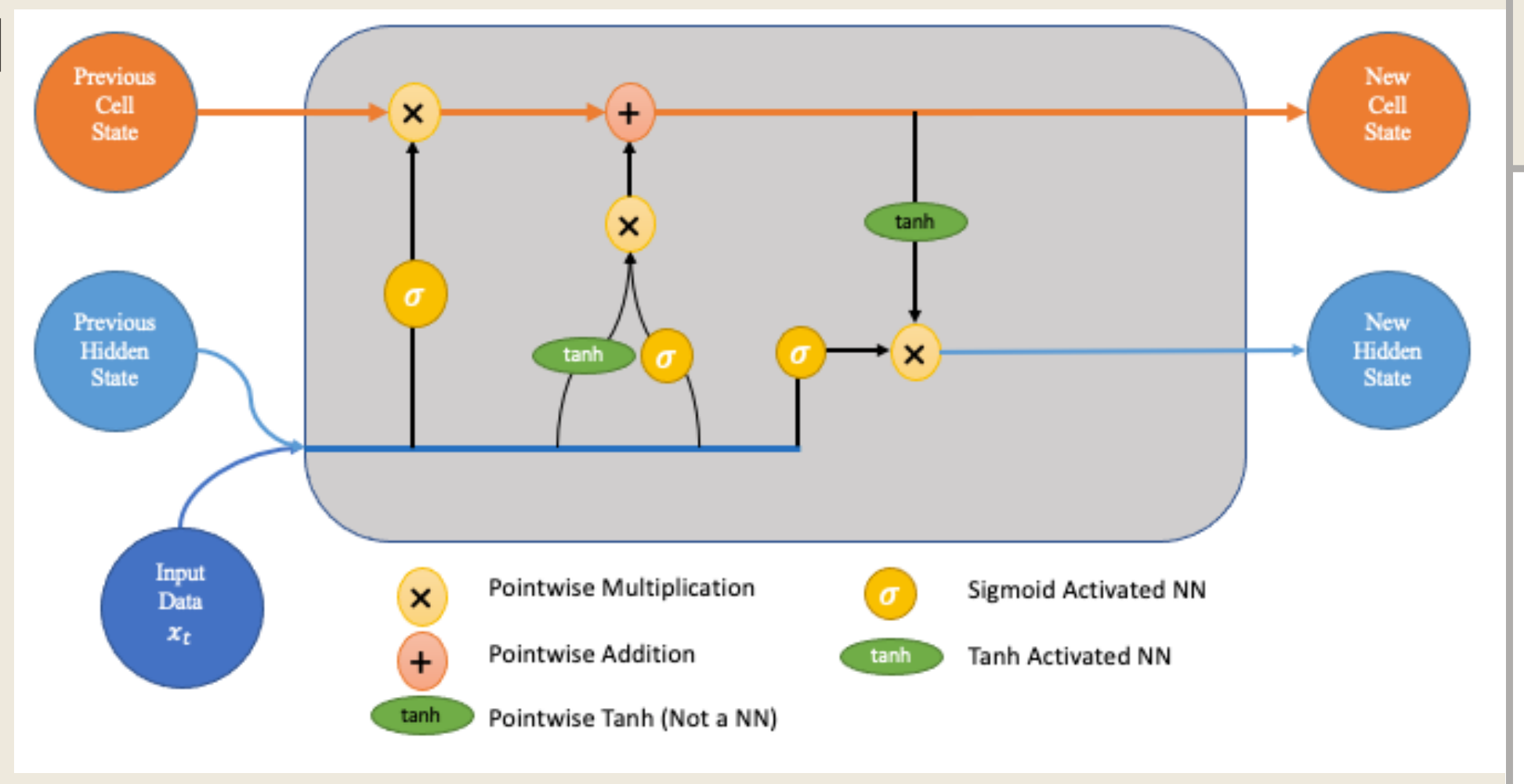
03. Background

MDP vs POMDP:

- The agent does not know the current state fully - Partially Observable Markov Decision Process (POMDP).
- Knows the current state - Markov Decision Process (MDP)
- Methods make POMDP to MDP

LSTM (lONG SHORT TERM MEMORY):

- Adding LSTM changes DQN to Deep Recurrent Q Network (DRQN), stores hidden states.
- In training, it uses Backpropagation Through Time (BPTT), uses the last n time steps.



***LSTM Cell*[5]**

FRAME STACKING:

- Input becomes 4 frames combined instead of 1.

Related literature

- [1]Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 1). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.
- [2] Madhuparna Bhowmik. Deep recurrent q-network, Jan 2019.
- [3] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2017.
- [4] Cĺ ement Romac and Vincent B́ eraud. Deep recurrent q-learning vs deep q-learning on a simple partially observable markov decision process with minecraft, 2019.
- [5] Rian Dolphin. LSTM networks: A detailed explanation. Medium, Dec 2021.
- [6] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, second edition, 2018.

04. Methodology

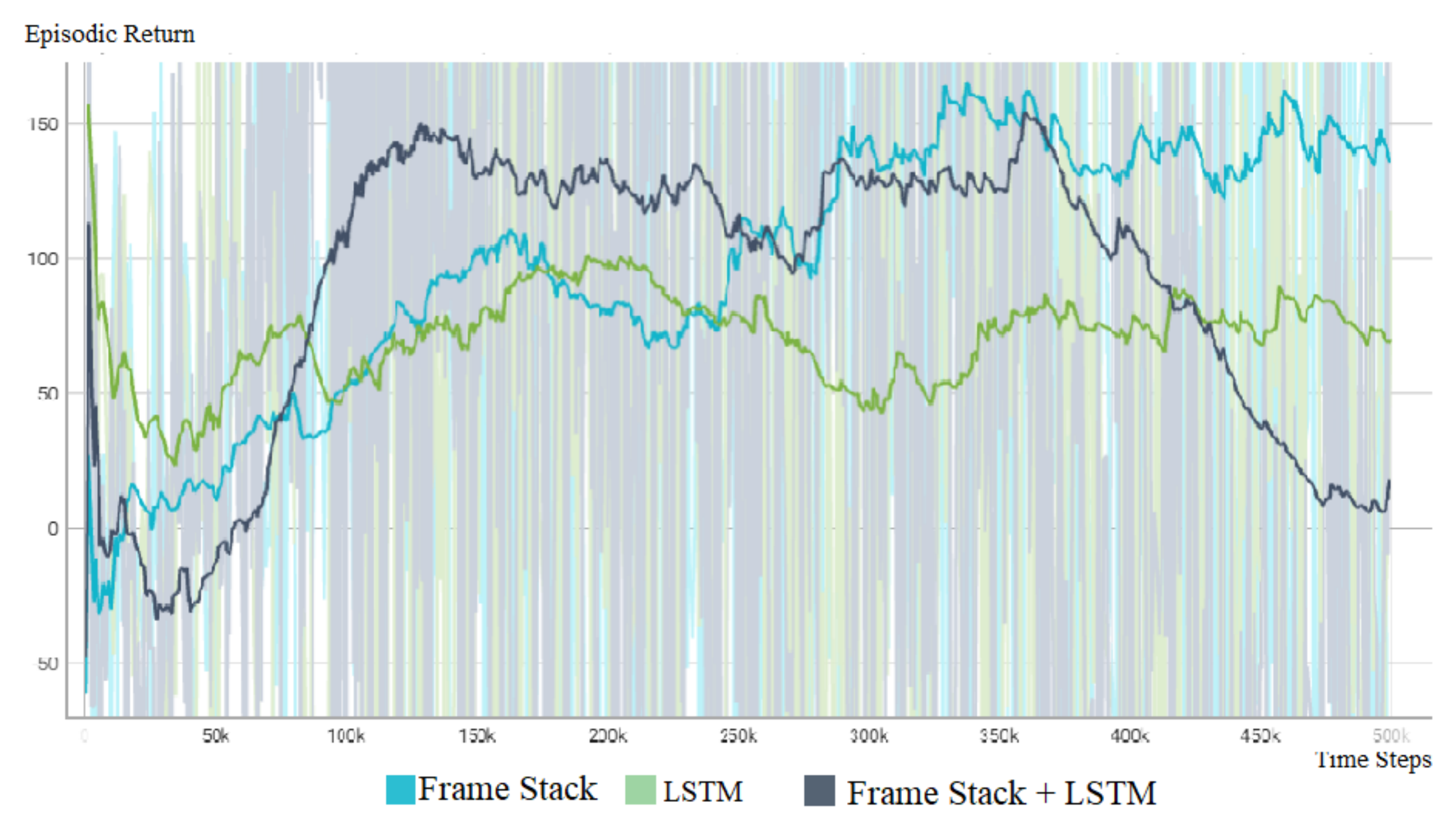
- Setting up CARLA in Delft Blue[1] (supercomputer of TU Delft) and connecting it with the Gymnasium framework
- Create 3 agents, all of which use DQN as a base. One uses frame stacking, one uses LSTM and one uses both.
- Check agents in a game-like environment, then train them in a Carla environment.
- Run each agent 10 times in 3 different tracks to test the robustness and collect data.

05. Limitations

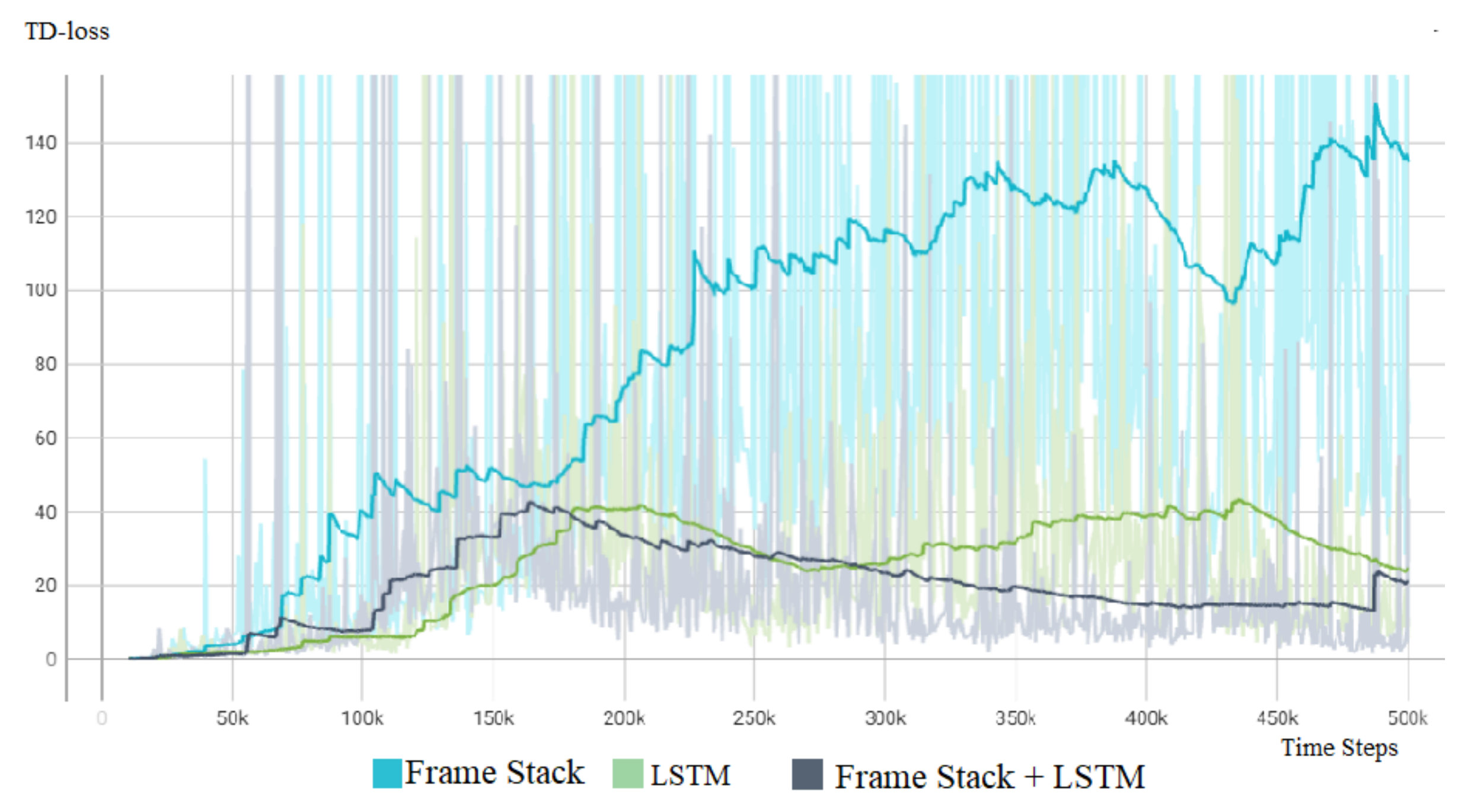
- Running the simulation is resource heavy.
- Not possible to capture real-life experiences.

06. Results in Training Phase

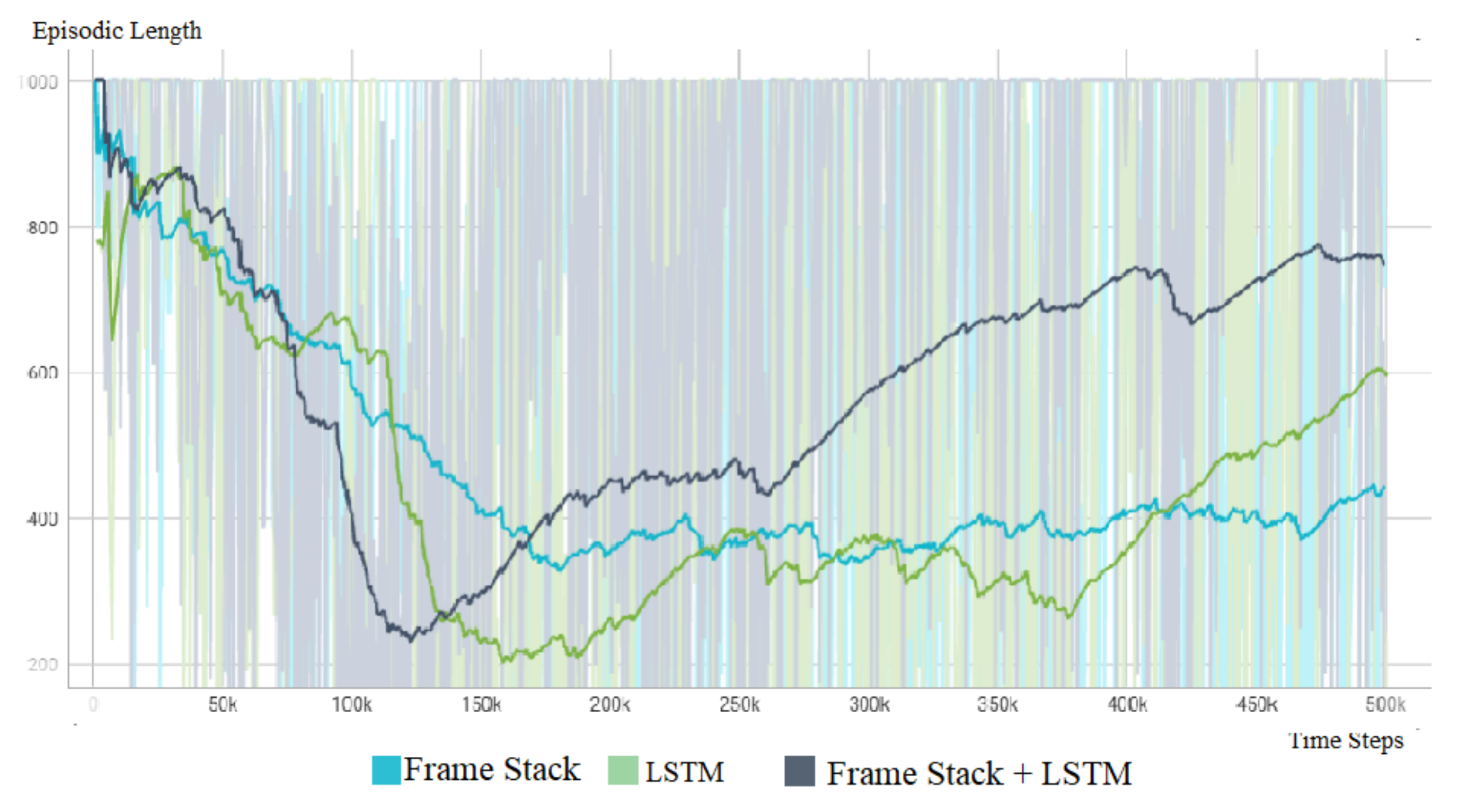
- DRQN with frame stacking started learning earlier, experienced catastrophic forgetting between 360k and 500k time steps, and finished with the lowest episodic return.
- All agents were similarly efficient at updating their Q values at the beginning.
- DQN with frame stacking finished training faster than other agents with the highest episodic return.



Smoothed Episodic returns in the training process of agents



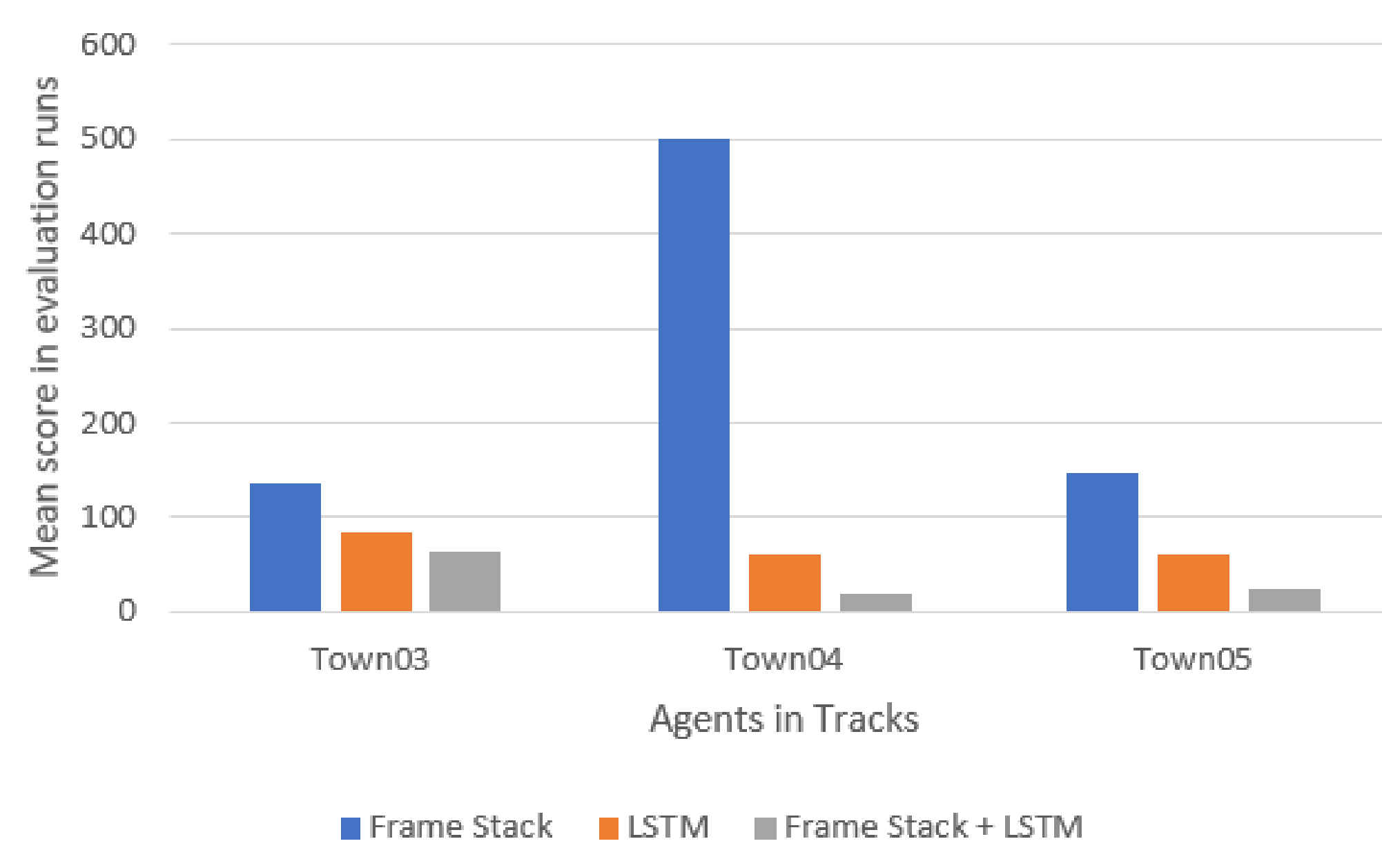
Smoothed temporal difference (TD) loss in the training process of agents



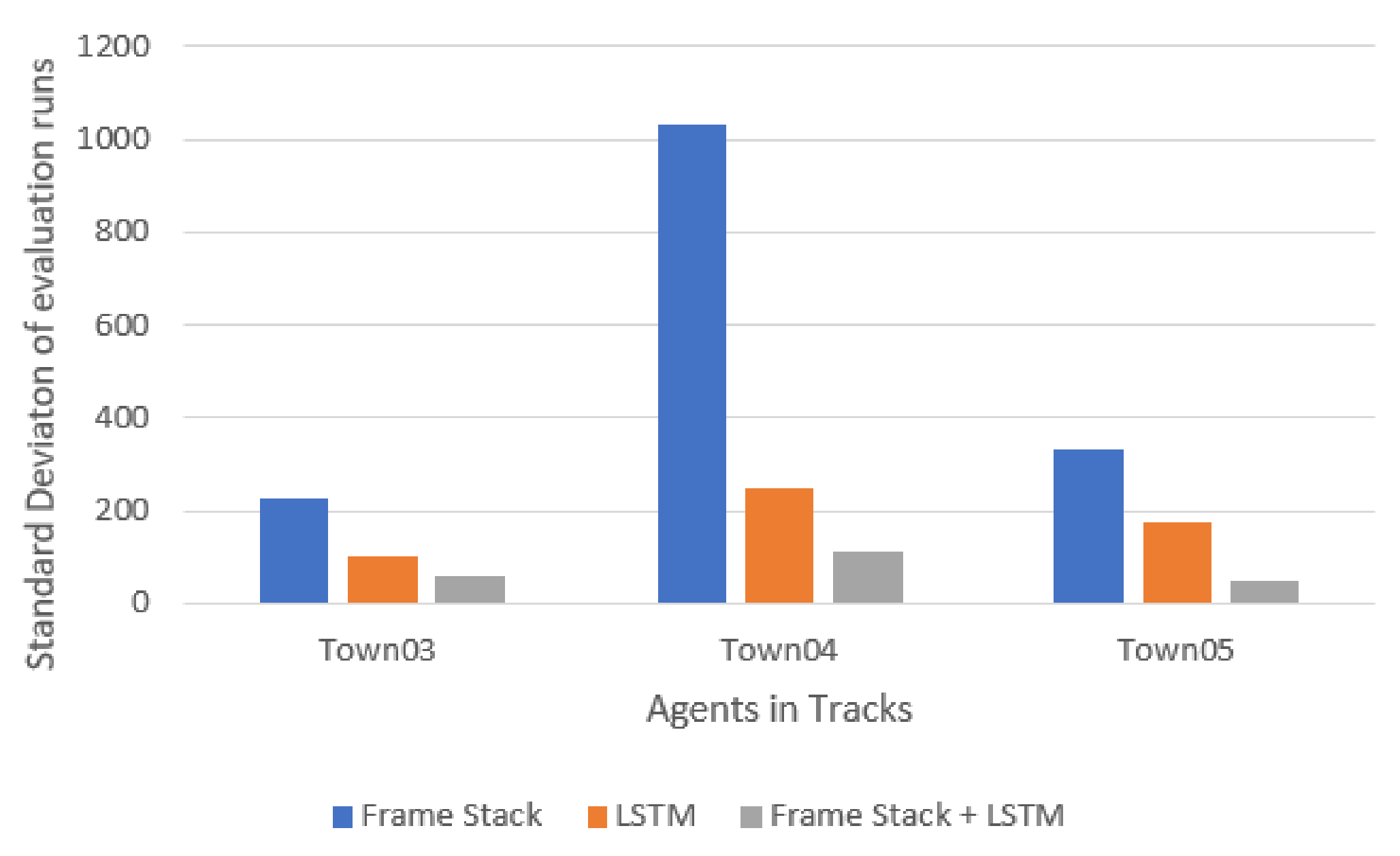
Smoothed Episodic lengths in the training process of agents

07. Results in Evaluation Phase

- DQN with frame stacking: High performance, low robustness
- DRQN: lower performance, higher robustness.
- DRQN with frame stacking: performed worse, overall robustness is considered compromised.



Mean episodic return of 3 agents 10 episodes in 3 different tracks



The standard deviation of 3 agents 10 episodes in 3 different tracks

08. Conclusion

- LSTM is not suitable for Car driving agents, as much as frame stacking is.
- LSTM is more robust which is a trade-off between performance and robustness.
- When LSTM and frame stacking is combined, even though it learns at earlier time steps, it creates an unstable agent.
- LSTM adds complexity to the agent's network, which increases the need for resources and increases the time required to finish training.