Dynamic Car-following Model Calibration with Deep Reinforcement Learning

Htet Naing[1]

I. INTRODUCTION

This document contains supplementary materials of our paper, *"Dynamic Car-following Model Calibration with Deep Reinforcement Learning"*. The paper aims to provide all model training parameters (or hyperparameters) that were used in calibrating the parameters of Intelligent Driver Model (IDM). Two DRL architectures, Deep Q-Network (DQN) [1], [2] and Deep Deterministic Policy Gradient (DDPG) [3], are adapted for this calibration task. All the DRL models used in our paper and the car-following simulation environment were implemented in Python language with the help of Pytorch [4]. The baseline model, Genetic Algorithm (GA), was implemented using a Python library distributed on Pypi [5].

II. IDM Related Parameters

Before the DRL-based calibration process begins, the default set of IDM parameters λ is inferred from the existing studies [6]–[8], and used as the initial starting point for the calibration task. Since these λ values can only range within their valid intervals, their respective parameter bounds are inferred from the training datasets, the road speed limit of the study area, and most importantly, the acceleration/deceleration bounds are based on a systematic study conducted in [9].

The default parameter values are then defined as: [33.3, 1.6, 2, 0.73, 1.67, 4] for $[v_0, T, gap_{jam}, acc_{max}, dacc_{max}, \delta]$, respectively. These values were initialized at the beginning of the training process of both DQN and DDPG. Accordingly, the lower (LB_λ) and upper bounds (UB_λ) are:

- set to = [10, 33.3333] m/s for maximum desired velocity v_0.
- set to [0.3, 6] s for desired time headway T.
- set to [1, 5] m for jam spacing distance gap_{jam}.
- set to [0.28, 3.41] m/s^2 for maximum desired acceleration acc_{max}.
- set to [0.47, 3.41] m/s^2 for maximum desired deceleration $dacc_{max}$.
- set to [0, 10] for exponent parameter δ.

III. DRL Training Hyperparamters

The training hyperparameters are referred to [10] and [11] for DDPG and [1], [12] for DQN.

A. DQN Training Hyperparameters

The architecture of the DQN network used in our paper is shown in Figure 1. DQN is used to calibrate three IDM paramters: acc_{max}, T, δ in our paper. The hyperparamters used for DQN is defined as follows.

- Number of training episodes E is set to 150.
- Learning rate (α) used for Deep Q-Network parameter update during the training process is set to 0.0005.
- Discount factor (γ) used to discount future rewards is set to 0.95.
- Replay memory size used to store transition experiences is set to 100000.
- Learning start point is set to 20000, i.e., the agent only starts learning when there are 20000 transitions in the memory.
- Batch size for transition samples (B) used for training Q-Network parameters is set to 256.
- The exploration probability ϵ is set to 1.0 initially and is decreased after each simulation step by the amount, $\epsilon_{dec} = 0.00001$ until it reaches the $\epsilon_{min} = 0.01$.
- Target network update frequency is set to once per 30 training episodes.
- Loss function is set to mean squared error loss.
- Optimization algorithm is set to Adam.

1) Model Selection for Testing: As for testing the DQN agent, the best Q-Network saved after the training episode that achieves the highest score was used for dynamic calibration. Similarly, for the best set of IDM parameters generated after that particular episode was also used for both static calibration and parameter initialization for dynamic calibration.

B. DDPG Training Hyperparameters

The architecture of the DDPG network used in our paper is shown in Figure 2. DDPG is used to calibrate all IDM parameters: $v_0, T, gap_{jam}, acc_{max}, dacc_{max}, \delta$ in our paper. The hyperparamters used for DDPG is defined as follows.

- Number of training episodes E is set to 150.
- Learning rate (α) used for the parameter update of both actor and critic networks during the training process is set to 0.0005.
- Discount factor (γ) used to discount future rewards is set to 0.95.
- Replay memory size used to store transition experiences is set to 100000.

[1]Htet Naing is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore

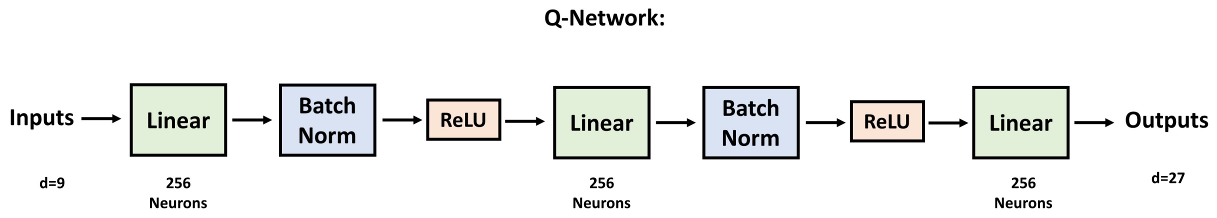

Fig. 1. DQN architecture used for car-following model calibration

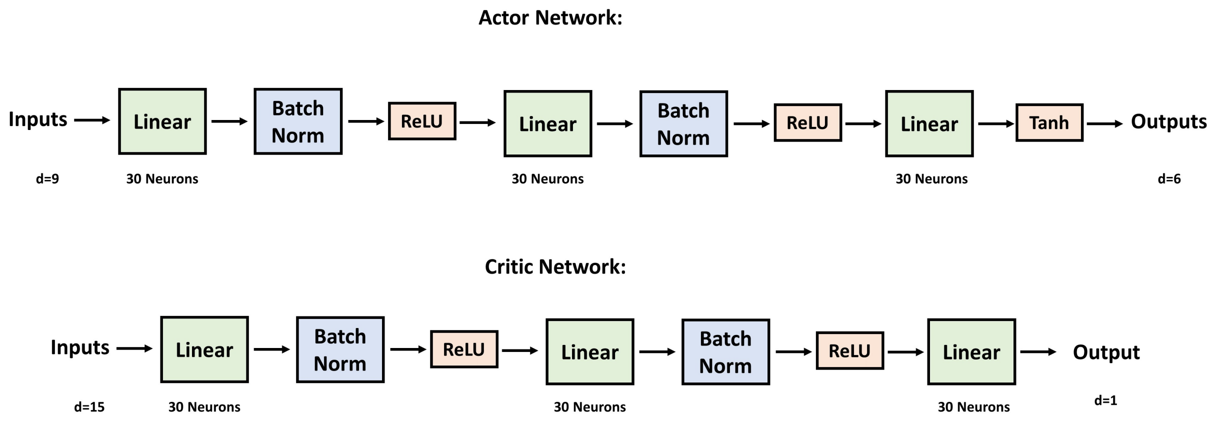

Fig. 2. DDPG architecture used for car-following model calibration

- Learning start point is set to 20000, i.e., the agent only starts learning when there are 20000 transitions in the memory.
- Batch size for transition samples (B) used for training actor and critic network parameters is set to 256.
- OU-Noise parameters added for action exploration are set to 0.15 and 2 for θ_{OU} and σ_{OU} respectively. Unlike DQN exploration, the OU-Noise used here is not decayed throughout the training process. Nonetheless, based on the training performance scores, the DDPG agent seems to converge to optimal behaviour without decaying this noise.
- Soft target update rate τ used for updating target network parameters is set to 0.001.
- Loss function is set to mean squared error loss for critic network and negative average Q-values, for actor network.
- Optimization algorithm is set to Adam.

1) Model Selection for Testing: As for testing the DDPG agent, the last actor network saved after the end of training process was used for dynamic calibration. The average IDM parameter values taken from the last 30 training episodes was used for both static calibration and parameter initialization for dynamic calibration.

IV. Genetic Algorithm Training Parameters

The GA is used to calibrate all IDM parameters: $v_0, T, gap_{jam}, acc_{max}, dacc_{max}, \delta$ in our paper. The GA training parameters used for static (offline) calibration of the IDM parameters are described below. The above-mentioned IDM parameter bounds were also applied to GA as constraints during the optimisation process.

- Maximum number of generation is set to 150.
- Population size is set to 50.
- Mutation probability is set to 0.3.
- Elite ratio (for selecting the number of elites out of a given population) is set to 0.01.
- Crossover probability is set to 0.5.
- Crossover method is set to be uniform crossover.
- Proportion of the number of parents in the population is set to 0.3.
- Stopping criterion is set as follows. When the solution for the last 10 generations has not improved anymore, the GA is terminated.

All training follower-leader trajectories were used to calculate the fitness score at each generation. The spacing root mean squared error for all simulated trajectories was used as the minimization objective subject to the IDM parameter bounds. Despite 150 generations set for optimisation, the early convergence was observed at the generation 20 after which the algorithm was terminated. Another optimisation run without any early termination introduced was also tested, however, the final solution generated after completing 150 generations, was observed to underperform that of the first run with early termination.

1) Model Selection for Testing: As for testing the GA performance, the final set of parameters generated after the algorithm convergence was selected for static calibration.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] M. (a.k.a. Facebook), "Pytorch," https://pytorch.org, 2021.

[5] R. M. Solgi, "Genetic algorithm," https://pypi.org/project/geneticalgorithm/, 2020.

[6] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[7] Z. Mo, R. Shi, and X. Di, "A physics-informed deep learning paradigm for car-following models," *Transportation research part C: emerging technologies*, vol. 130, p. 103240, 2021.

[8] M. Treiber, A. Kesting, and D. Helbing, "Delays, inaccuracies and anticipation in microscopic traffic models," *Physica A: Statistical Mechanics and its Applications*, vol. 360, no. 1, pp. 71–88, 2006.

[9] P. S. Bokare and A. K. Maurya, "Acceleration-deceleration behaviour of various vehicle types," *Transportation research procedia*, vol. 25, pp. 4733–4749, 2017.

[10] F. Hart, O. Okhrin, and M. Treiber, "Formulation and validation of a car-following model based on deep reinforcement learning," *arXiv preprint arXiv:2109.14268*, 2021.

[11] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation research part C: emerging technologies*, vol. 97, pp. 348–368, 2018.

[12] X. Tang, B. Gong, Y. Yu, H. Yao, Y. Li, H. Xie, and X. Wang, "Joint modeling of dense and incomplete trajectories for citywide traffic volume inference," in *The World Wide Web Conference*, 2019, pp. 1806–1817.