Learning to Calibrate Hybrid Hyperparameters: a Study on Traffic Simulation

Wanpeng Xu
Fordham University
New York, USA
wanpeng653@gmail.com

Hua Wei
New Jersey Institute of Technology
Newark, USA
hua.wei@njit.edu

ABSTRACT

Traffic simulation is an important computational technique that models the behavior and interactions of vehicles, pedestrians, and infrastructure in a transportation system. Calibration, which involves adjusting simulation parameters to match real-world data, is a key challenge in traffic simulation. Traffic simulators involve multiple models with hybrid hyperparameters, which could be either categorical or continuous. In this paper, we present *CHy*^{2}, an approach that generates a set of hyperparameters for simulator calibration using generative adversarial imitation learning. *CHy*^{2} learns to mimic expert behavior models by rewarding hyperparameters that deceive a discriminator trained to classify policy-generated and expert trajectories. Specifically, we propose a hybrid architecture of actor-critic algorithms to handle the hybrid choices between hyperparameters. Experimental results show that *CHy*^{2} outperforms previous methods in calibrating traffic simulators.

KEYWORDS

Reinforcement learning, traffic simulation, model calibration

ACM Reference Format:
Wanpeng Xu and Hua Wei. 2023. Learning to Calibrate Hybrid Hyperparameters: a Study on Traffic Simulation. In *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '23), June 21–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3573900.3591113

1 INTRODUCTION

Traffic simulation is a computational technique that models the behavior and interactions of vehicles, pedestrians, and infrastructure in a transportation system. It has been widely used to analyze and predict the performance of transportation systems, including traffic congestion [16], travel time [7], and safety [5].

One of the main challenges in traffic simulation is calibration, which involves adjusting simulation parameters to match real-world data [3]. Traffic simulators typically include multiple models, such as microscopic simulators [9, 17], which include car-following models (representing vehicle acceleration and deceleration), lane-changing models (representing vehicle movements between lanes), junction models (representing vehicle interactions at intersections), and others, where each model could contain *hybrid hyperparameters*, *i.e.*, the hyperparameters could be either categorical or continuous.

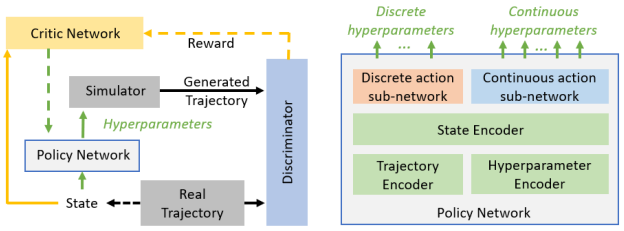

Figure 1: Proposed *CHy*^{2} Approach. Left: The overall framework of *CHy*^{2} includes three components: policy network, critic network, and discriminator network. Right: The detailed network architecture of the policy network in *CHy*^{2}, which has separate sub-networks for discrete actions and continuous actions. Best viewed in color.

In this paper, we present *CHy*^{2}, an approach that can **c**alibrate a set of **hy**brid **hy**perparameters for the simulator. Given a set of real-world observations (also called expert trajectories), *CHy*^{2} learns to mimic the expert behavior models under the framework of generative adversarial imitation learning (GAIL) [6], which learns a policy that can perform expert-like hyperparameters by rewarding the hyperparameters for deceiving a discriminator trained to classify between policy-generated and expert trajectories. Specifically, for hybrid hyperparameters, we propose a hybrid architecture of actor-critic algorithms for the policy network to deal with the hybrid choices between hyperparameters. It is based on the original architecture of PPO algorithms [12] but contains multiple parallel sub-policy networks instead of one to solve hyperparameter selection respectively and has one global critic network to update the sub-policy networks. We show that *CHy*^{2} outperforms previous methods in simulating realistic trajectories and output hyperparameters more precisely for the microscopic traffic simulators.

2 RELATED WORK

Heuristic-driven Calibration This is the traditional and most common method of calibration, where experts pre-define the form of the model, and hyperparameters are adjusted until an acceptable fit is achieved between the model outputs and observed data [1, 2, 10]. Although this method is simple and straightforward to implement, it can be time-consuming and requires a high level of expertise in model development and data analysis.

Data-driven Calibration Data-driven methods are widely used for modeling complex systems, relying on statistical or machine

SIGSIM-PADS '23, June 21–23, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0030-9/23/06…$15.00
https://doi.org/10.1145/3573900.3591113

learning techniques to learn relationships between inputs and outputs from available data [6, 13, 15]. They have the advantage of being automated and requiring minimal user input, making them a popular choice for many applications. However, these methods can require large amounts of data. In situations where data is limited or biased, data-driven methods may not capture the complexity of the model accurately. Additionally, if existing simulators do not support machine learning models as internal models, these methods may not be applicable.

3 PRELIMINARY

In our problem, the hyperparameters before each round of simulation are controlled by an agent. At each round t, agent i observes from the environment as its state o_i^t. Given the vehicle position, the goal of the agent is to give the optimal action a (i.e., which hyperparameters to set), so that the similarity between simulated trajectories and real trajectories can be maximized.

Simulator Calibration as a Markov Decision Process. We can formally model the simulator calibration task by a Markov Decision Process (MDP), defined by a tuple $\Gamma =< \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma >$, where $\mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma$ are the sets of states, transition probability functions, joint actions, reward functions and a discount factor respectively:

• $\mathcal{S}$: At each time step t, agent observes the state $s^t \in \mathcal{S}$. Our state includes the current hyperparameter settings and the ground truth trajectories.

• $\mathcal{A}$. An agent's action set $\mathcal{A}$ is defined as a group of hyperparameters. In the traffic simulator, $\mathcal{A}$ is mostly pre-defined, i.e., the candidate hyperparameters are set to be chosen from a finite set.

• $\mathcal{P}$: At time step t, the agent takes an action $a^t \in \mathcal{A}$, which induces a transition according to the state transition function: $\mathcal{P}(s^{t+1}|s^t, a^t) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

• $\mathcal{R}$: In a Markov Process, the reward an agent i obtains obtains rewards r^t at time t by a reward function $\mathcal{R}(s^t, a^t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Considering our problem definition, we do not know the similarity between the simulated trajectories and observed trajectories, and thus need to learn the reward function.

• γ: Each agent i aims to maximize its total discounted reward $G_i^t = \Sigma_{k=t}^{E} \gamma^{k-t} r_i^t$ from time step t onwards, where the discount factor $\gamma \in [0, 1]$ controls the importance of immediate rewards versus future rewards, and E is the length of an episode that controls the total rounds of simulation. The termination of a simulation round t is conditioned on the reward r^t smaller than a threshold ϵ. if the current simulation fails to achieve above ϵ, the simulation with the current hyperparameter setting would terminate early.

The policy π of the agent has a corresponding policy function that gives the action probabilities $\pi(a|s)$ conditioned on the observation s, when acting according to that policy π.

4 METHOD

4.1 Generative Adversarial Imitation Learning

In this paper, we follow the framework similar to GAIL [6] due to its scalability to the multi-agent scenario and previous success in learning human driver models [8]. GAIL formulates imitation learning as the problem of learning policy to perform expert-like behavior by rewarding it for "deceiving" a classifier trained to discriminate between policy-generated and expert state-action pairs. For a neural network classifier $\mathcal{D}_\psi$ parameterized by ψ, the GAIL objective is given by $max_\psi\, min_\theta\, \mathcal{L}(\psi, \theta)$ where $\mathcal{L}(\psi, \theta)$ is :

$$\begin{aligned}\mathcal{L}(\psi, \theta) = \mathbb{E}_{(s,a)\sim\tau\in\mathcal{T}_E} \log \mathcal{D}_\psi(s, a) \\ + \mathbb{E}_{(s,a)\sim\tau\in\mathcal{T}_G} \log(1 - \mathcal{D}_\psi(s, a)) - \beta H(\pi_\theta)\end{aligned} \tag{1}$$

where $\mathcal{T}_E$ and $\mathcal{T}_G$ are respectively the expert trajectories and the generated trajectories from the interactions of policy π_θ with the simulation environment, $H(\pi_\theta)$ is an entropy regularization term.

• *Learning* ψ: When training $\mathcal{D}_\psi$, Equation (1) can simply be set as a sigmoid cross entropy where positive samples are from $\mathcal{T}_E$ and negative samples are from $\mathcal{T}_G$. Then optimizing ψ can be easily done with gradient ascent.

• *Learning* θ: The simulator is an integration of physical rules, control policies and randomness and thus its parameterization is assumed to be unknown. Therefore, given $\mathcal{T}_G$ generated by π_θ in the simulator, Equation (1) is non-differentiable w.r.t θ. In order to learn π_θ, GAIL optimizes through reinforcement learning, with a surrogate reward function formulated from Equation (1) as:

$$\tilde{r}(s^t, a^t; \psi) = -\log(1 - \mathcal{D}_\psi(s^t, a^t)) \tag{2}$$

Here, $\tilde{r}(s^t, a^t; \psi)$ can be perceived to be useful in driving π_θ into regions of the state-action space at time t similar to those explored by π^E. The optimization of θ is optimized via algorithms with actor-critic style including TRPO [11] and PPO [12], which usually have one actor network and one critic network, and the critic network is used to compute the gradient of the parameters of the actor network.

4.2 Policy Network for Hybrid Action Space

The hyperparameters in the simulator could be either discrete or continuous. In microscopic traffic simulators, the hyperparameters could be the parameters for Lane Changing Models (LCM), Car-Following Models (CFM), and Junction Models (JM), where each model has multiple choices with hybrid types. For example, the CFM can be chosen from several candidate models as discrete hyperparameters and each candidate model for CFM has continuous hyperparameters like minimum gap when standing, maximum acceleration ability of vehicles, and maximum deceleration, *etc.*. This makes the action space for the policy a class of discrete-continuous hybrid action spaces.

To tackle the hybrid action space problem, we propose an architecture for hybrid action spaces (shown in Figure 1) that contains two parallel actor networks (or even more for general hierarchical action spaces). The parallel actors perform discrete selection and continuous selection separately: one discrete actor network learns a stochastic policy π_{θ_d} to select the discrete action a and one continuous actor network learns a stochastic policy π_{θ_c} to choose the continuous parameters. The complete action to execute is the selected action paired with the chosen continuous hyperparameter a_c and discrete hyperparameter a_d. The two actor networks share a state encoder to encode the state information.

There is a single critic network in the hybrid actor-critic architecture, which works as an estimator of the state-value function $V(s)$. In our architecture, the state-value function $V(s)$ is used for

computing a variance-reduced advantage function estimator $\hat{A}$. We follow the implementation in Equation (3), which runs the policy for T rounds and computes the estimator $\hat{A}_t$ using the collected samples as where $t \in [0, T]$ is the round index and E is much less than the length of an episode:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{E-t} r_{t+1} \tag{3}$$

With a critic network providing an estimation of the advantage function, the hybrid actor-critic architecture is flexible in the choice of the policy optimization method. The only requirement is that the optimization method should have an actor-critic style and updates stochastic policies with the advantage function provided by the critic. Although the complete action to execute a is decided by both of the actors, the discrete actor and the continuous actor are updated separately by their respective update rules at each round. The update rules for the discrete policy π_{θ_d} and the continuous policy π_{θ_c} network could follow policy gradient methods such as TRPO [11] or PPO [12].

4.3 Training and Implementation

In this paper, the policy network consists of a trajectory encoder, a hyperparameter encoder followed by a state encoder, and two sub-networks. The trajectory encoder is parameterized by a Transformer [14] to encode the observed trajectories, other networks including the hyperparameter encoder, state encoder, and sub-networks, are implemented by two-layer fully connected networks with 32 units for all the hidden layers. The policy network takes the state s as input and outputs the distribution parameters for a Normal distribution, and the action a will be sampled from this distribution. The policy networks are optimized via TRPO [11]. We aim to optimize the hyperparameters microscopic traffic simulator, specifically three models in the simulator: Lane Changing Models (LCM), Car-Following Models (CFM), and Junction Models (JM). Their detailed hyperparameters can be found in Table 1 with the full list described in the official documentation of SUMO [1], and the policy takes the state as input and outputs an action a (i.e., hyperparameters). For the discriminator network, each driving point is embedded to a 10-dimensional latent space and fed into a two-layer fully connected layer.

5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 Datasets. The proposed framework is demonstrated and evaluated on the dataset from two different microscopic traffic simulators focusing on the diagnostics of vehicle movements. We use CityFlow [17] as the target simulator to be calibrated, and SUMO [9] as the source simulator where we have the ground truth of simulator hyperparameters. We also use a real-world open-sourced dataset from Los Angeles [18] to evaluate the differences between generated trajectories and real-world observed trajectories.

5.1.2 Baselines. We compare our model with the following two categories of methods: heuristic-driven methods and data-driven methods.For Heuristic-driven methods, we use the default models of simulator CityFlow [9] [17].

[1]Full hyperparameters can be found in https://bit.ly/sumo-models.

Table 1: Hyperparameters considered in this paper. The complete list can be found in the official documentation of SUMO.

Category	Type	Description
Which CFM?	Discrete	7 candidates: Krauss, KraussOrig1, ...
Which LCM?	Discrete	2 candidates: LC2013, SL2015
CFM-related	Continuous	31 hyperparameters, such as minGap, accel, decel, ...
LCM-related	Continuous	27 hyperparameters, such as lcStrategic, lcCooperative, ...
JM-related	Continuous	12 hyperparameters, such as jmCrossingGap, impatience, ...

• **Random Search (*RS*)** [2]: The parameters are chosen when they generate the most similar trajectories to expert demonstrations after a finite number of trials of random selecting parameters.
• **Tabu Search (*TS*)** [10]: Tabu search chooses the neighbors of the current set of parameters for each trial. If the new CFM generates better trajectories, this set of parameters is kept in the Tabu list.
• **Behavioral Cloning (*BC*)** [13] is a traditional imitation learning method. It directly learns the state-action mapping in a supervised manner.
• **Generative Adversarial Imitation Learning (*GAIL*)** is a GAN-like framework [6], with a generator controlling the policy of the agent, and a discriminator containing a classifier for the agent indicating how far the generated state sequences are from that of the demonstrations.

5.1.3 Evaluation Metrics. Following existing studies [4, 8, 18], to measure the error between learned policy against the expert policy, we measure the position (ps) and the travel time (tm) of vehicles between generated dense trajectories and expert dense trajectories, which are defined as: $RMSE_{pos} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{M}\sum_{i=1}^{m}(l_i^t - \hat{l}_i^t)^2}$, $RMSE_{time} = \sqrt{\frac{1}{M}\sum_{i=1}^{m}(d_i - \hat{d}_i)^2}$ where T is the total simulation time, M is the total number of vehicles, l_i^t and $\hat{l}_i^t$ are the position of i-th vehicle at time t in the expert trajectories and in the generated trajectories relatively, d_i and $\hat{d}_i$ are the travel time of vehicle i in expert trajectories and generated trajectories respectively. If we know the ground truth of hyperparameters, the inference accuracy is evaluated by Acc@1 for discrete hyperparameters and the root mean square error (RMSE) for continuous hyperparameters. Acc@1 (Accuracy at top-1) is a common evaluation metric used to measure the accuracy of predictions, calculated by dividing the number of correct predictions in top-1 predictions by the total number of predictions.

5.2 Results

In this section, the performance of the proposed method is analyzed based on two evaluation criteria: inference accuracy and computational cost.

5.2.1 Inference Accuracy. One of the primary objectives of simulator calibration is to make the output of the simulator close to the observations. In traffic simulation, we consider the root mean square error (RMSE) between the observed vehicle trajectory and the simulated vehicle trajectory. Table 2 shows the simulation performance of the baseline heuristic-driven and data-driven methods and our proposed method (CHy^2) in the dataset generated by SUMO

Table 2: Performance w.r.t Relative Mean Squared Error (RMSE) of time (in seconds) and position (in kilometers) on SUMO dataset and real-world dataset in Los Angeles (LA). The lower the better.

	SUMO		LA	
	time (s)	pos (km)	time (s)	pos (km)
RS	4.4278	0.6154	4.3932	0.4696
TS	9.7713	0.4142	4.2928	0.8775
BC	6.7349	0.3254	5.5595	0.7299
GAIL	1.3611	**0.0345**	0.5297	0.0631
CHy^2	**1.1648**	0.0347	**0.4394**	**0.0578**

and the real-world dataset in Los Angeles [15]. We can see that *CHy*^{2} achieves the best performance with smaller error under most cases, indicating its effectiveness in simulating realistic trajectories. It is worth mentioning that data-driven methods like *GAIL* do not assume the form of the simulation models like heuristic-driven methods and *CHy*^{2} do, which replaces the inner models of simulators completely with a machine learning model. It requires the simulator to import machine learning models, but sadly most traffic simulators do not support it now.

The primary objective of simulator calibration is to infer the values of the model parameters. From the application perspective of model-based diagnostics, this objective corresponds to inferring the true underlying degradation parameters. Therefore, we compare the estimated hyperparameters with the ground truth. Table 3 shows the inference performance of the baseline methods and *CHy*^{2} under the SUMO dataset where we know the groundtruth hyperparameters. With the lowest RMSE, the policy obtained with *CHy*^{2} shows the best overall performance in both datasets. The *RS* and *TS* model yields worse overall performance, which highlights the limitations of heuristic-given methods. It is worth mentioning that unlike traditional data-driven methods, which replace the inner models of simulators completely with a machine learning model our framework does not replace the inner models but improves the search for better hyperparameters. This makes our method more flexible and more applicable to existing simulators.

Table 3: The inference accuracy w.r.t. Acc@1 and RMSE on the hyperparameters under the dataset from SUMO. For Acc@1, the higher the better; for RMSE, the lower the better.

	Lane Changing		Car-Following		Junction
	Acc@1	RMSE	Acc@1	RMSE	RMSE
RS	0.678	7.7936	0.1625	9.9117	5.2100
TS	0.6820	7.4297	0.2431	9.8905	4.9488
BC	0.7615	7.1506	0.2973	9.4845	5.0302
GAIL	0.8469	6.3103	0.3473	9.4046	4.3802
CHy^2	**0.8615**	**5.3763**	**0.5390**	**8.7899**	**3.9425**

5.2.2 Computational Cost. One crucial aspect of the proposed method is the ability to perform calibration in real-time which is a crucial requirement for real applications. Therefore, we evaluate the time required to perform inference of the model parameters at deployment. Table 4 reports the average times required to calibrate a single sample and the total training time with the five methods. In terms of deployment computational cost, the proposed method

Table 4: Average time required for inference of a single sample with heuristic-driven and data-driven approaches in seconds for LA dataset.

Method	*RS*	*TS*	*BC*	*GAIL*	*CHy*^{2}
Deployment time (s)	6	5.1	2.01e-2	2.10e-2	2.10e-2

is 300 times faster compared to the *TS* and *RS*. This deployment speed is comparable to the *BC* and *GAIL* models as both methods only require a forward pass over a deep neural network.

6 CONCLUSION

In this paper, we present *CHy*^{2}, an approach that generates a set of hyperparameters for simulator calibration using generative adversarial imitation learning. With a hybrid architecture of actor-critic algorithms, *CHy*^{2} is able to handle the hybrid choices between hyperparameters. Preliminary results show that *CHy*^{2} outperforms previous methods in calibrating traffic simulators. In the future, we will conduct more experiments on the robustness of our proposed method under stochastic simulators and noisy data, and investigate the scalability of the proposed method.

ACKNOWLEDGMENTS

The work was supported in part by NSF award #2153311. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Richard Arsenault, Annie Poulin, Pascal Côté, and François Brissette. 2014. Comparison of stochastic optimization algorithms in hydrological model calibration. *Journal of Hydrologic Engineering* 19, 7 (2014), 1374–1384.
[2] Johannes Asamer, Henk J van Zuylen, and Bernhard Heilmann. 2013. Calibrating car-following parameters for snowy road conditions in the microscopic traffic simulator VISSIM. *IET Intelligent Transport Systems* 7, 1 (2013).
[3] Keith Beven and Andrew Binley. 1992. The future of distributed models: model calibration and uncertainty prediction. *Hydrological processes* 6, 3 (1992), 279–298.
[4] Raunak P Bhattacharyya, Derek J Phillips, et al. 2018. Multi-agent imitation learning for driving simulation. In *IROS 2018*. IEEE.
[5] Douglas Gettman and Larry Head. 2003. Surrogate safety measures from traffic simulation models. *Transportation Research Record* 1840, 1 (2003), 104–115.
[6] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *NeurIPS*.
[7] Natalia Ruiz Juri, Avinash Unnikrishnan, and S Travis Waller. 2007. Integrated traffic simulation–statistical analysis framework for online prediction of freeway travel time. *Transportation Research Record* 2039, 1 (2007), 24–31.
[8] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE.
[9] Pablo Alvarez Lopez, Michael Behrisch, et al. 2018. Microscopic traffic simulation using sumo. In *ITSC 2018*. IEEE.
[10] Carolina Osorio and Vincenzo Punzo. 2019. Efficient calibration of microscopic car-following models for large-scale stochastic network simulators. *TR-B* (2019).
[11] John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. In *ICML*.
[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv* (2017).
[13] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *IJCAI*.
[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. 2017. Attention is all you need. *NeurIPS 2017* 30 (2017).
[15] Hua Wei, Chacha Chen, Chang Liu, Guanjie Zheng, and Zhenhui Li. 2021. Learning to simulate on sparse trajectory data. In *ECML-PKDD 2020*. Springer, 530–545.
[16] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. 2019. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117* (2019).
[17] Huichu Zhang, Siyuan Feng, et al. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. In *TheWebConf 2019*.
[18] G Zheng, H Liu, K Xu, and Z Li. 2020. Learning to Simulate Vehicle Trajectories from Demonstrations. In *ICDE*.