Towards Efficient Personalized Driver Behavior Modeling with Machine Unlearning

Song, Qun; Tan, Rui; Wang, Jianping

DOI
10.1145/3576914.3587489

Publication date
2023

Document Version
Final published version

Published in
Proceedings of 2023 Cyber-Physical Systems and Internet-of-Things Week, CPS-IoT Week 2023 - Workshops

Towards Efficient Personalized Driver Behavior Modeling with Machine Unlearning

Qun Song*
q.song-1@tudelft.nl
Delft University of Technology
The Netherlands

Rui Tan
tanrui@ntu.edu.sg
Nanyang Technological University
Singapore

Jianping Wang
jianwang@cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

ABSTRACT

Driver Behavior Modeling (DBM) aims to predict and model human driving behaviors, which is typically incorporated into the Advanced Driver Assistance System to enhance transportation safety and improve driving experience. Inverse reinforcement learning (IRL) is a prevailing DBM technique with the goal of modeling the driving policy by recovering an unknown internal reward function from human driver demonstrations. However, the latest IRL-based design is inefficient due to the laborious manual feature engineering processes. Besides, the reward function usually experiences increased prediction errors when deployed for unseen vehicles. In this paper, we propose a novel deep learning-based reward function for IRL-based DBM with efficient model personalization via machine unlearning. We evaluate our approach on a highway simulation constructed using the realistic human driving dataset NGSIM. We deploy our approach on both a server GPU and an embedded GPU. The evaluation results show that our approach achieves a higher prediction accuracy compared with the latest IRL-based DBM approach that uses a weighted sum of trajectory features as the reward function. Our model personalization method obtains the highest accuracy and lowest latency compared with the baselines.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

Driver behavior modeling, inverse reinforcement learning, neural network, model personalization, machine unlearning

ACM Reference Format:
Qun Song, Rui Tan, and Jianping Wang. 2023. Towards Efficient Personalized Driver Behavior Modeling with Machine Unlearning. In *Cyber-Physical Systems and Internet of Things Week 2023 (CPS-IoT Week Workshops '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3576914.3587489

*Part of this work was done while the author was at Nanyang Technological University.

CPS-IoT Week Workshops '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0049-1/23/05.
https://doi.org/10.1145/3576914.3587489

1 INTRODUCTION

Driver Behavior Modeling (DBM) [6] aims to model and predict human driving behaviors such as driving maneuvers, driver intent, vehicle state, etc. DBM models are incorporated as essential components of the Advanced Driver Assistance System (ADAS) to improve transportation safety and passenger experience [1]. With the advancement in artificial intelligence, data-driven approaches such as imitation learning (IL) [8], which aims to learn the human-like driving policy from enormous human demonstration data, have become the prevailing DBM method. Among IL techniques, inverse reinforcement learning (IRL) [7] assumes that a human driver follows an optimal internal driving policy with an unknown reward function for making driving decisions. The goal of IRL is to recover the reward function and imitate human driving from realistic human demonstrations. IRL has become an important tool for DBM due to its capability in providing accurate driver intent prediction and good generalizability to different driving situations.

However, the existing research on IRL-based DBM is inadequate for the following reasons. First, existing studies (e.g., [7, 10]) adopt a linear combination of manually designed human trajectory features, such as travel efficiency, ride comfort, risk aversion, etc., as the reward function. This is inefficient because the manually engineered trajectory features require laborious engineering efforts and can be biased according to the designer's scope of knowledge. Second, when deploying the reward function on new vehicles that are unseen in the training stage, the prediction error usually increases due to the changes in driver preferences and driving conditions.

To address the aforementioned challenges, this paper proposes a novel deep neural network (DNN)-based reward function for IRL-based DBM with efficient model personalization via machine unlearning. To eliminate the laborious manual feature engineering effort, we use a ConvLSTM-based RewardNet as the reward function to automatically extract the useful trajectory features for driver behavior evaluation. To efficiently adapt the general RewardNet from the training stage to an unseen individual vehicle, we propose a novel model personalization method based on machine unlearning, which can efficiently adapt a well-trained factory model to a new user using the observed inconsistency between RewardNet predictions and human executed actions. We evaluate our approach on a highway simulation environment constructed using the realistic vehicle trajectory dataset NGSIM [2]. The evaluation results show that the RewardNet achieves lower prediction error compared with the reward function in the latest IRL-based DBM approach [7] that is defined as the weighted sum of manually selected trajectory features. The evaluation results also show that our model personalization method obtains the highest accuracy among all the baseline approaches including weighted sum, fine-tuning,

training-from-scratch, and few-shot learning methods. To measure the model personalization overhead, we deploy our method on a server GPU as well as a prevailing embedded GPU platform for autonomous vehicles, i.e., NVIDIA Jetson AGX Xavier. The experiment results show that our method obtains the lowest average model personalization latency among all the baselines.

The main contributions of this paper are as follows:

- We propose a DNN-based RewardNet as the reward function for IRL-based DBM that outperforms the state-of-the-art approach relying on manual feature engineering.
- We design an efficient model personalization method for adapting the RewardNet from the training stage to new unseen vehicles using the machine unlearning technique.
- We evaluate our approach on a highway simulation constructed on a realistic human driving dataset. The evaluation results show that our approach has the highest personalization accuracy and lowest latency on a server GPU and an NVIDIA Jetson embedded GPU among all the baselines, including the weighted sum, fine-tuning, training-from-scratch, and few-shot learning methods

The rest of this paper is organized as follows. Section 2 gives the background of this paper and reviews related work. Section 3 states the problem and introduces the NGSIM-based driver behavior modeling simulation environment. Section 4 presents our DNN-based RewardNet and our efficient model personalization method using machine unlearning. Section 5 evaluates our approach. Section 6 concludes this paper.

2 BACKGROUND AND RELATED WORK

This section presents the background of our study and reviews the related work.

2.1 Driver Behavior Modeling with Inverse Reinforcement Learning

Driver Behavior Modeling (DBM) [6] aims to understand, predict, and simulate driver behaviors. *Rule-based* DBM simulates driver behaviors by creating a set of rules defining how drivers respond to different stimuli. Besides the laborious manual effort to design the model, rule-based DBM cannot adapt to environmental changes and is unable to learn from new data over time. *Machine learning-based* DBM has received increasing research attention because of its efficiency in automatically extracting useful features from enormous human driving trajectories. *Imitation learning* (IL) is one of the state-of-the-art machine learning-based DBM methods, which aims to learn a policy that mimics human behaviors from real human driving demonstrations [8]. As an imitation learning technique, *behavioral cloning* (BC) fits a model from a dataset of expert state-action pairs using supervised learning. However, it is prone to the problem of cascading errors [9] under the scenes that are under-represented in the training data. With the generative adversarial network (GAN) technique, *generative adversarial imitation learning* (GAIL) trains an agent to capture the behavior of a human driver and a discriminator to distinguish the agent's outputs and the real-world trajectories [9]. *Inverse reinforcement learning* (IRL) [7] assumes that a human driver uses an optimal internal driving policy to make driving decisions. The goal of IRL is to recover the unknown reward function of the driving policy from the observed human driving behaviors. IRL-based DBM takes into account the driving context and driver preference, which demonstrates good generalizability to unseen scenes. Thus, in this work, we consider IRL-based DBM.

2.2 Machine Learning Model Personalization

In general, model personalization belongs to *domain adaptation*, where a machine learning model is adapted from the source domain defined by the training dataset to a specific user or task in the target domain [9]. *Transfer learning* [11] is a prevailing method to address domain shifts, which retrains the model with new target-domain samples. However, existing transfer learning techniques require enormous target-domain data samples and incur high post-deployment overhead. *Few-shot learning* [12] aims to adapt the model from the source domain to the target domain using a small amount of data. However, since few-shot learning is usually trained on a limited amount of data, it suffers from the issues of overfitting and decreased accuracy on unseen data. In this work, our goal is to retain the useful knowledge learned from the factory production stage, while fine-tuning the model with the newly collected data of an individual driver. Specifically, we aim to improve the fine-tuning performance by rectifying the inappropriate model's behaviors indicated by the inconsistency observed between the driver's action and the model's output. Thus, we consider the machine unlearning technique. Machine unlearning aims to delete the influence of some particular training data points on the target model's parameters [4]. For non-adaptive machine learning algorithms (e.g., naïve Bayes), it is possible to know exactly how a training data point contributes to the model parameters and delete this contribution [4]. However, for adaptive machine learning algorithms (e.g., neural networks), the model's parameter update depends on any previous iteration. Thus, it is difficult to know how each data point affects the target model. Retraining the model from scratch on the remaining data can completely unlearn a data point, but incurs impractical compute overhead. Therefore, the existing approaches focus on more efficient retraining-based unlearning. The work in [3] trains multiple submodels on different non-overlapping training data shards and stores many intermediate states of each submodel. Then, data point unlearning is achieved by only retraining the submodel related to the data to be unlearned from the last intermediate state of the submodel that has not yet been trained on the data. This method incurs excessive storage overhead and may result in weak learners. In this work, we design a more compute- and memory-efficient machine unlearning algorithm that, by modifying the loss function during the fine-tuning stage, rectifies the model's predictions inconsistent with human behaviors.

3 CONSTRUCTING DBM MODEL

3.1 Problem Statement

In IRL-based DBM, it is assumed that human driving follows the below behavioral procedures [7]. *Trajectory generation:* given a random traffic scene, a driver generates multiple candidate trajectories in mind. *Trajectory evaluation:* for each candidate trajectory, the driver anticipates the outcome of executing it. In particular, the driver uses an internal reward function to evaluate each trajectory

and each trajectory is assigned an execution probability based on its reward value. *Trajectory selection:* the driver executes the final trajectory according to the trajectory probability distribution. In IRL-based DBM, it is usually assumed that human drivers execute the candidate trajectories by following the Boltzmann distribution [7], i.e., the probability of executing a candidate trajectory is exponential to the reward of that trajectory:

$$P(\zeta \mid \boldsymbol{\theta}) = \frac{e^{R(\zeta;\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})} \approx \frac{e^{R(\zeta;\boldsymbol{\theta})}}{\sum_{i=1}^{M} e^{R(\zeta_i;\boldsymbol{\theta})}}, \tag{1}$$

where ζ is a candidate trajectory, $R(\cdot;\boldsymbol{\theta})$ is the reward function with parameters $\boldsymbol{\theta}$, and Z is the integration of all possible trajectories. Since $Z(\theta)$ is intractable due to high dimensionality, it can be approximated using a group of M candidate trajectories $\{\zeta_1, \zeta_2, \ldots, \zeta_M\}$. In this work, we generate candidate trajectories in the NGSIM-based simulation as described in Section 3.2 and Section 3.3. To derive the execution probability distribution of the candidate trajectories, the key element is the reward function. After the form of the reward function $R(\cdot;\boldsymbol{\theta})$ is decided, e.g., weighted sum or neural network, we need to find out the parameters $\boldsymbol{\theta}$ of the reward function such that the driving policy characterized by the reward function matches human's demonstrations. The previous work [7] assumes a linear reward function, which is defined as the weighted sum of the selected features, including travel efficiency, ride comfort, risk aversion, and interaction. In this work, we apply a DNN-based reward function to avoid the laborious manual feature engineering, which will be discussed in Section 4 shortly. Formally, the problem can be stated as follows: given a set of human driving trajectories $\mathcal{D} = \{\zeta_1^*, \zeta_2^*, \ldots, \zeta_N^*\}$, derive the DNN-based reward function with parameters $\boldsymbol{\theta}$ that generates a driving policy to maximize the output reward values of human demonstrations among all the candidate trajectories. To fit the reward function parameters, we use the maximum-entropy IRL algorithm that is commonly adopted in existing works [19]. However, the derived reward function may suffer from increased error when deploying for unseen vehicles due to domain shifts caused by distinct driving preferences, changes of environment, etc. Thus, a related problem is that, given an unseen vehicle, to adapt the reward function parameters $\boldsymbol{\theta}$ to reduce the prediction error on this new vehicle. We aim to maximize the reduced error while minimizing the model personalization overhead.

3.2 Highway Simulation Environment

In this work, we build the highway simulation environment based on the open-source Next-Generation Simulation (NGSIM) dataset [2]. Specifically, the NGSIM dataset contains detailed vehicle trajectory data collected at 10Hz from the US Highway 101 and Interstate 80 Freeway for 45 minutes each. The US Highway 101 covered in NGSIM is about 640 meters long and has five main lanes and one auxiliary lane with on-ramp and off-ramp. The Interstate 80 covered in NGSIM is about 500 meters long and has six freeway lanes, including one high-occupancy vehicle lane and an on-ramp.

We now describe our highway simulation environment built based on the NGSIM dataset. We construct the multi-lane highway as described in the dataset. The vehicles are spawned on the road at the time instances recorded in the dataset. The vehicles' dynamics follow the kinematic bicycle model [13]. At each driving scene, we select an ego vehicle to be observed and consider its surrounding vehicles within a perception range. By default, the surrounding vehicles follow their original trajectories in the dataset. If the gap between a vehicle and its front vehicle is smaller than a pre-defined threshold, this vehicle's longitudinal responses will be overridden by the parametric car-following intelligent driver model (IDM) [15] to give an acceleration command in order to reach a target velocity as well as maintain a minimum safety distance with respect to the front vehicle. The lateral response is characterized by the steering wheel angle command. Specifically, the lateral speed command is obtained using a proportional controller with gain $\mathrm{KP}_{lateral} = -\frac{1}{3}(1/s)$ to steer the vehicle to follow the center of the lane that it is running on. The lateral speed command will be converted to a heading reference. The heading reference is then used to generate a heading rate command using a proportional controller with gain $\mathrm{KP}_{heading} = \frac{1}{0.2}(1/s)$. This heading rate command will be converted to the steering angle command according to the bicycle model. Then, the bicycle model is used to propagate the state of the overridden vehicle taking the acceleration and steering angle commands as input. The state contains the vehicle's position, velocity, and heading.

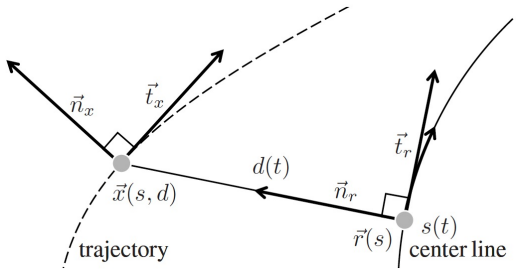

Figure 1: Trajectory generation in the Frenét-frame.

3.3 Trajectory Generation

This section describes how we generate the candidate trajectories to be evaluated by the reward function. In the simulation, the trajectory generation is performed on the Frenét space [5]. The transformation is depicted in Fig. 1. Specifically, the Frenét coordinates are built upon a reference path, which is the central line of the highway lane in our case. The position of a vehicle is described using the variables s and d, where s coordinate is the distance along the reference path and d coordinate represents the lateral displacement to the reference path. The translation of the coordinates on the Frenét space to the Cartesian space is given as: $\vec{x}(s(t), d(t)) = \vec{r}(s(t)) + d(t)\vec{n}_r(s(t))$. Transforming the calculations on the Cartesian coordinates to the Frenét frame improves the trajectory computation efficiency by employing simpler computations for the longitudinal and lateral trajectories separately [17]. Given the initial and target states of the vehicle, we aim to generate a trajectory that has minimized jerks along the path. Thus, the cost function is given by the integral of the square of jerk: $J_t(p(t)) \triangleq \int_{t_0}^{t_1} \dddot{p}^2(\tau)d\tau$, where p can be d or s. It has been proven in [14] that the solution to any jerk optimization problem can be expressed using a polynomial

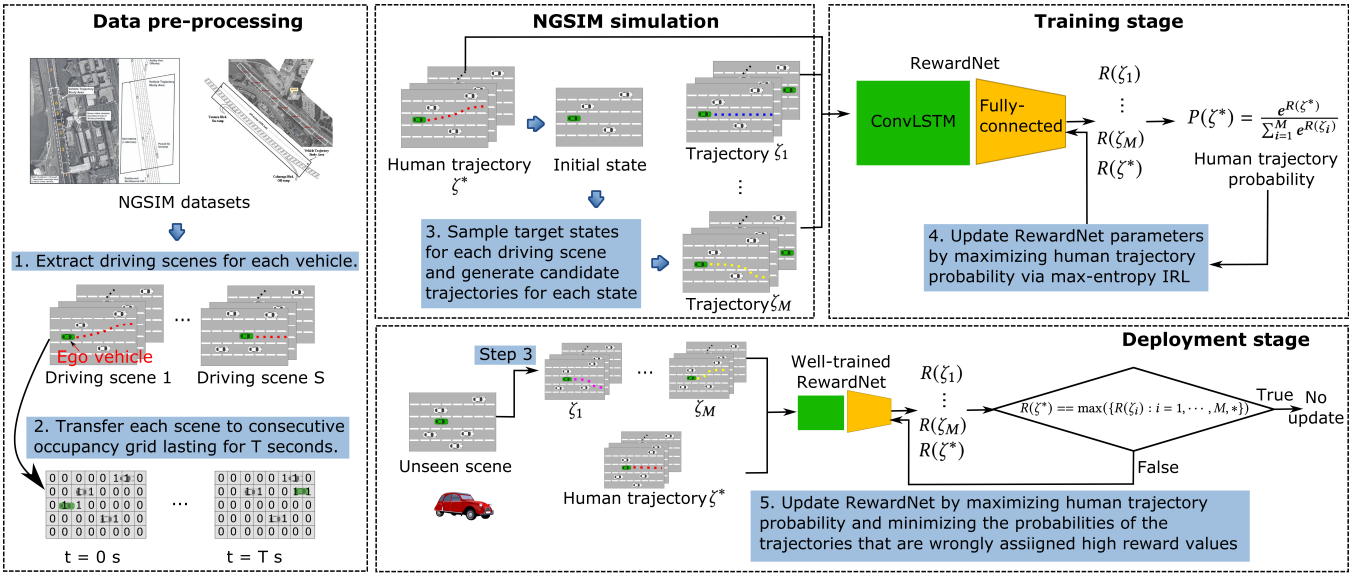

Figure 2: Illustration of the proposed approach.

function: $p(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \alpha_3 t^3 + \alpha_4 t^4 + \alpha_5 t^5$. Therefore, given the initial state $[X_s, Y_s, V_{xs}, V_{ys}, A_{xs}, A_{ys}]$ at $t = 0$ and the target state $[X_e, Y_e, V_{xe}, V_{ye}, A_{xe}, A_{ye}]$ at $t = T$, as well as the required time T to reach the target state, the coefficients of the polynomial functions can be decided. Thus, the location of the vehicle at each time instance is derived and the trajectory is generated.

We determine the target sampling space for the candidate trajectories as follows. Given an initial state of the ego vehicle, we consider two variables for the target state, i.e., the target longitudinal velocity V_{ye} and target lateral position X_e. That is, we set the remaining target space variables to zero. We sample ten possible target longitudinal velocities from the range $[V_{ys} - 5, V_{ys} + 5]$. The ego vehicle may plan to make decisions of changing lane to left, to right, and keeping in lane. Thus, the target lateral positions can be $[X_{l_s-1}, X_{l_s}, X_{l_s+1}]$, where l_s is the lane ID for the initial state and X_{l_s} is the center lateral position of lane l_s. Note that when the vehicle is initially in the leftmost lane, the target lateral positions can only be $[X_{l_s}, X_{l_s+1}]$. A similar rule is applied when the vehicle is in the rightmost lane. Therefore, for each initial driving scene, we generate 20 or 30 possible trajectories to be evaluated. In particular, we rule out the trajectories that end with collision or off-road situations in the simulation.

4 APPROACH

This section presents the proposed approach, as illustrated in Fig. 2, to personalized DBM with machine unlearning.

4.1 Data Preprocessing

For each vehicle on the highway, we extract multiple driving scenes each lasting for T seconds. In this work, we set $T = 5s$. Each driving scene contains the trajectory of the ego vehicle and the trajectories of its surrounding vehicles. In total, each driving scene consists of 50 time steps with the sampling rate of NGSIM simulation set to 10 Hz. For each time step of a driving scene, we encode the driving trajectories into an occupancy grid. If a grid cell has a vehicle on it, this grid cell is assigned a value of one. Otherwise, it is assigned a value of zero. Note that a vehicle (e.g., a truck) may occupy multiple grid cells. The ego vehicle appears in the longitudinal center of the grid. The lateral position depends on the lane that it is on. The width of the occupancy grid equals the width of the highway (all lanes) in the NGSIM simulation. We assume that the perception range of the ego vehicle is 150 feet. Thus, the grid covers an area of 72 feet in width and 300 feet in length. Each grid cell has a width of 6 feet and a length of 15 feet. Each input data sample to the DNN-based reward function presented in Section 4.2 consists of T-second consecutive driving scene snapshots in the form of occupancy grids.

4.2 DNN-based Reward Function for IRL-based DBM

Instead of manually engineering the trajectory features and determining the parameters of the reward function with respect to the selected features, we train a DNN-based *RewardNet* to automatically extract useful features from the driving trajectories and output the reward values. The input data samples to the RewardNet are the raw trajectories preprocessed by following the procedures in Section 4.1. Since driving trajectories are time-series data, we adopt the ConvLSTM [18] architecture in the RewardNet to better extract the time correlation embedded. The ConvLSTM module of the RewardNet has one layer with 16 hidden channels and kernel size of 3. The ConvLSTM module is then cascaded by two fully connected layers with 128 neurons each. The output layer of the RewardNet has one neuron which generates the reward value. The goal of training the RewardNet is to maximize the probability of human trajectory $P(\zeta^* \mid \boldsymbol{\theta})$. Thus, with the preprocessed training data samples, the RewardNet is optimized via the maximum entropy IRL algorithm [19] using the following loss function:

$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta^* \in \mathcal{D}} -\log P(\zeta^* \mid \boldsymbol{\theta})$. The loss function is minimized using gradient descent.

4.3 Efficient RewardNet Personalization via Machine Unlearning

After the training stage, we derive a well-trained *general* RewardNet that captures the human driving behaviors observed from the training dataset. However, this general RewardNet may suffer from increased error when deployed for unseen driver's trajectories. To address this problem, we develop an efficient model personalization method using machine unlearning for adapting the general RewardNet on an individual vehicle. The workflow is as follows. During the deployment stage on a new individual vehicle, we use the general RewardNet to evaluate the human trajectory as well as the generated candidate trajectories in each driving scene. Given a driving scene, if human trajectory obtains the highest reward value, we do not update the general RewardNet. Otherwise, the scenario where the output reward value for the human trajectory is not the highest means that the driving policy characterized by the RewardNet predicts actions inconsistent with the human driver's. Under such scenarios, we unlearn the general RewardNet on this driving scene. Specifically, we aim to adapt the general RewardNet by maximizing the probability of human trajectory $P(\zeta^* \mid \boldsymbol{\theta})$ and minimizing the probabilities of the trajectories that are wrongly assigned high reward values. We consider the top K trajectories excluding the human trajectories that have the highest reward values output by the RewardNet. Thus, we still apply the maximum-entropy IRL algorithm and use the following loss function for RewardNet personalization:

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{\zeta^* \in \mathcal{D}} (-\log P(\zeta^* \mid \boldsymbol{\theta}) + \sum_{\zeta_k \in \mathcal{K}} \log P(\zeta_k \mid \boldsymbol{\theta})), \qquad (2)$$

where $\mathcal{K}$ contains the K candidate trajectories that obtain the highest reward values, excluding the human trajectory. This updated loss function is optimized using gradient descent.

5 EVALUATION

This section presents the evaluation of our approach.

5.1 Evaluation Settings

We conduct the experiments on our computing server. The server has 10-core Intel Core i9-7900X 3.30GHz CPU and runs Ubuntu 18.04. The server is equipped with NVIDIA GeForce RTX 2080 Ti 11GB graphics processing units (GPUs). All our codes are implemented using Python 3.7. To measure the model personalization latency, we deploy our approach on both the computing server and an NVIDIA Jetson AGX Xavier, which is a prevailing embedded GPU platform equipped with an octa-core 2.26GHz ARM CPU, a 512-core Volta GPU, and 16GB RAM.

To train the *general* RewardNet, we randomly select 100 vehicles from the NGSIM dataset and follow the procedures described in Section 3.3 to generate candidate trajectories. We train the RewardNet until the training loss converges. Then, we randomly pick another 100 vehicles that are not included in the training data for testing. To evaluate the prediction accuracy, we compute the *average displacement error* (ADE) of the predicted trajectories with respect to the human trajectories as the evaluation metric. Specifically, we compare the general RewardNet with the reward function in [7] which is defined as the weighted sum of manually designed trajectory features. Note that the work in [7] is one of the latest research for IRL-based DBM. We use the same training and testing data as our approach for comparisons.

Table 1: Training and post-deployment average displacement errors (ADEs) of our approach and the work in [7] that uses the weighted sum of manually selected features as the reward function.

	RewardNet (ours)	Weighted sum [7]
Training ADE (m)	0.00	2.12
Post-deploy ADE (m)	0.53	2.68

To evaluate the model personalization performance, we consider the following baselines. The *fine-tuning* method only maximizes the human trajectory probability for the new driving scenes where the trajectories predicted by the general RewardNet are inconsistent with human's demonstrations; The *training-from-scratch* method trains a new RewardNet for each individual vehicle. In particular, for each vehicle, we train its RewardNet for 200 iterations. The *general* method directly applies the general RewardNet on each individual vehicle, ignoring the inconsistent observations. Lastly, we follow the description in [12] and implement the *few-shot regression* method, which is a few-shot learning method that utilizes Bayesian meta-learning with deep kernels for cross-domain adaption using limited training samples. To evaluate the model personalization accuracy, we randomly select 150 vehicles that are different from the ones used for training and testing the general RewardNet. For each individual vehicle, we adapt the models using 30 driving scenes and evaluate the performance using 20 driving scenes.

5.2 Evaluation Results

The training and post-deployment ADEs for the general RewardNet and weighted-sum approach [7] are presented in Table 1. We can see that the RewardNet achieves much lower ADEs in both the training and post-deployment stages. This is because the work in [7] requires manually selecting trajectory features that are relevant to driving behavior. However, the selected features can be biased according to designer's experience and may not be the optimal ones. In comparison, our RewardNet employs a DNN to automatically extract useful features from raw trajectories, which is more efficient. However, from the results, we can also observe an increased ADE on new unseen vehicles during the post-deployment stage compared with the training stage for both RewardNet and weighted sum approaches. In what follows, we present the performance of our unlearning-based model personalization method in reducing this domain shift error.

The average ADEs for different model personalization methods are presented in Fig. 3. It is shown that the unlearning and fine-tuning methods have similar prediction accuracy, with unlearning approach achieving slightly lower ADE as shown in Table 2. However, the unlearning method also has lower model personalization latency as presented in Table 2. The ADE for the general RewardNet without personalization is 0.529 m, which is in accordance with

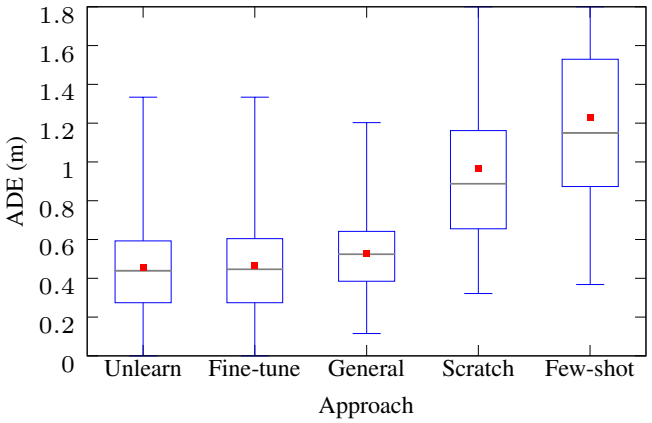

Figure 3: ADEs for different model personalization approaches. Note that whiskers represent the minimum and maximum; box represents the 25th and 75th percentiles; red dot represents the mean; gray bar represents the median.

Table 2: ADE and average model personalization latency on server GPU and embedded GPU.

	Unlearn	Fine-tune	Scratch	Few-shot
ADE (m)	**0.457**	0.466	0.964	1.14
Server latency (ms)	**298.98**	306.30	3389.49	302.81
Embedded latency (s)	**12.43**	12.80	139.57	12.55

the post-deployment ADE of RewardNet presented in Table 1. The ADE for the training-from-scratch method is higher than that of the general RewardNet. This is because the knowledge extracted from multiple human demonstrated trajectories in a wide range of driving scenes implemented can be transferred to a new unseen individual vehicle. However, the training-from-scratch approach can only learn from a limited amount of trajectories for each individual vehicle. Thus, the general knowledge cannot be utilized to improve the prediction accuracy. The few-shot regression approach has the highest ADE. The potential reason is that few-shot learning tends to overfit to a small amount of data used for model adaptation and is less generalizable to new unseen data points [16].

Lastly, we measure the average model personalization latency. From the results shown in Table 2, we can observe that our unlearning method obtains the lowest average execution latency among all the baselines on both the desktop-level GPU as well as the embedded GPU platform.

6 CONCLUSION

In this work, we proposed a novel deep learning-based reward function for IRL-based DBM with efficient model personalization via machine unlearning. In a highway simulation environment constructed based on realistic human driving datasets, we showed the satisfactory performance of the DNN-based RewardNet in human trajectory prediction. We also presented that the model personalization method utilizing the machine unlearning technique is effective in reducing the increased error during the deployment of RewardNet on unseen vehicles and is efficient running on both the server GPU and embedded GPU. The approach proposed in this paper can also be applied to other IL techniques for DBM, such as the GAIL.

ACKNOWLEDGMENTS

This research is supported in part by the National Research Foundation, Singapore and National University of Singapore through its National Satellite of Excellence in Trustworthy Software Systems (NSOE-TSS) office under the Trustworthy Computing for Secure Smart Nation Grant (TCSSNG) award no. NSOE-TSS2020-01, and in part by a project from Hong Kong Research Grant Council under GRF 11200220.

REFERENCES

[1] Najah AbuAli and Hatem Abou-Zeid. 2016. Driver behavior modeling: Developments and future directions. *International Journal of Vehicular Technology* 2016 (2016).
[2] Federal Highway Administration. 2020. Next Generation Simulation (NGSIM). https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.
[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *IEEE Symposium on Security and Privacy*. IEEE, 141–159.
[4] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy*. IEEE, 463–480.
[5] Manfredo P Do Carmo. 2016. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications.
[6] S Hamdar. 2012. Driver behavior modeling. *Handbook of Intelligent Vehicles* 33 (2012), 537–558.
[7] Zhiyu Huang, Jingda Wu, and Chen Lv. 2021. Driving Behavior Modeling Using Naturalistic Human Driving Data With Inverse Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* (2021).
[8] Parham M Kebria, Abbas Khosravi, Syed Moshfeq Salaken, and Saeid Nahavandi. 2019. Deep imitation learning for autonomous vehicles based on convolutional neural networks. *IEEE/CAA Journal of Automatica Sinica* 7, 1 (2019), 82–95.
[9] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium*. IEEE, 204–211.
[10] Mehmet F Ozkan, Abishek J Rocque, and Yao Ma. 2021. Inverse reinforcement learning based stochastic driver behavior learning. *IFAC-PapersOnLine* 54, 20 (2021), 882–888.
[11] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.
[12] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. 2020. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems* 33 (2020), 16108–16118.
[13] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. 2017. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?. In *2017 IEEE intelligent Vehicles Symposium*. IEEE, 812–818.
[14] Arata Takahashi, Takero Hongo, Yoshiki Ninomiya, and Gunji Sugimoto. 1989. Local path planning and motion control for agv in positioning. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems. The Autonomous Mobile Robots and Its Applications*. IEEE, 392–397.
[15] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E* 62, 2 (2000), 1805.
[16] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *Comput. Surveys* 53, 3 (2020), 1–34.
[17] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. 2010. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 987–993.
[18] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*. 802–810.
[19] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning.. In *AAAI Conference on Artificial Intelligence*, Vol. 8. Chicago, IL, USA, 1433–1438.