Continual Driving Policy Optimization with Closed-Loop Individualized Curricula

Haoyi Niu[1*], Yizhou Xu[1*], Xingjian Jiang[1], Jianming Hu[1,2†]
[1] Tsinghua University, Beijing, China
[2] Beijing National Research Center for Information Science and Technology, Beijing, China
{nhy22, xyz20, jiangxj21}@mails.tsinghua.edu.cn
hujm@mail.tsinghua.edu.cn [‡]

Abstract

The safety of autonomous vehicles (AV) has been a long-standing top concern, stemming from the absence of rare and safety-critical scenarios in the long-tail naturalistic driving distribution. To tackle this challenge, a surge of research in scenario-based autonomous driving has emerged, with a focus on generating high-risk driving scenarios and applying them to conduct safety-critical testing of AV models. However, limited work has been explored on the reuse of these extensive scenarios to iteratively improve AV models. Moreover, it remains intractable and challenging to filter through gigantic scenario libraries collected from other AV models with distinct behaviors, attempting to extract transferable information for current AV improvement. Therefore, we develop a continual driving policy optimization framework featuring **C**losed-**L**oop **I**ndividualized **C**urricula (CLIC), which we factorize into a set of standardized sub-modules for flexible implementation choices: *AV Evaluation*, *Scenario Selection*, and *AV Training*. CLIC frames AV Evaluation as a collision prediction task, where it estimates the chance of AV failures in these scenarios at each iteration. Subsequently, by re-sampling from historical scenarios based on these failure probabilities, CLIC tailors individualized curricula for downstream training, aligning them with the evaluated capability of AV. Accordingly, CLIC not only maximizes the utilization of the vast pre-collected scenario library for closed-loop driving policy optimization but also facilitates AV improvement by individualizing its training with more challenging cases out of those poorly organized scenarios. Experimental results clearly indicate that CLIC surpasses other curriculum-based training strategies, showing substantial improvement in managing risky scenarios, while still maintaining proficiency in handling simpler cases. Code is available at https://github.com/YizhouXu-THU/CLIC.

1 Introduction

With remarkable advancements in deep learning (DL) and deep reinforcement learning (DRL), autonomous driving has gained substantial interest from academia, industry, and the public. However, the deployment of autonomous vehicles (AV) in the real world has been significantly impeded by safety concerns. The primary crux sources from the distribution of naturalistic driving data (NDD), which exhibits a long-tailed pattern [Zhang *et al.*, 2023], leading to a severe data imbalance characterized by a scarcity of safety-critical scenarios. Naturally, solely relying on NDD would require training and testing AV for billions of miles [Kalra and Paddock, 2016] to ensure safety. Thereby, a repertoire of studies has emphasized the need to generate safety-critical scenarios [Wang *et al.*,

---

[*]Work done with equal contribution.

[†]Corresponding author.

2021a; Ding *et al.*, 2020; Feng *et al.*, 2020c,a,b; Sun *et al.*, 2021; Niu *et al.*, 2021; Lee *et al.*, 2020; Rempe *et al.*, 2022; Niu *et al.*, 2023] to address the data imbalance issue. This has led to the emergence of an exciting avenue coined as scenario-based autonomous driving [Nalic *et al.*, 2020; Li, 2020] that harbors several advantages against autonomous driving under naturalistic situations. Logged scenarios offer substantial reproducibility and flexibility for re-organization, avoiding repetitive computations during replay and allowing re-sampling of valuable scenarios at will. These advantageous properties hold great promise for accelerating the testing phase [Feng *et al.*, 2023; Zhao *et al.*, 2016, 2017; Huang *et al.*, 2017; Zhong *et al.*, 2021; Fremont *et al.*, 2020; Yang *et al.*, 2023] introduced with advanced re-sampling techniques over safety-critical scenarios.

However, limited research has explored the potential of utilizing the rich and diverse historical scenarios for closed-loop training of AV, rather than just for testing purposes. The main challenge is twofold: (1) *Scenario Transferability*: Pre-collected scenarios can vary significantly across different driving patterns, including those generated by overly timid and considerably aggressive AV models or human drivers. AV models trained directly on such scenarios may not always yield improvement due to the significant distribution inconsistency [Cao *et al.*, 2021]. (2) *Scenario Adaptation*: Unlike the fixed AV model used during the testing phase, closed-loop AV policy optimization inevitably involves dynamical improvement of the iterated AV model. Moreover, driving scenarios are widely acknowledged for their diversity in difficulty levels, so the AV model needs to be fed with carefully selected scenarios that align with their capabilities at each iteration. This highlights the need to offer individualized curricula comprising more challenging scenarios and less boring ones, tailored to fulfill the requirements of efficient and effective training at current stage.

To tackle these challenges, we introduce a novel framework of **Continual Driving Policy Optimization with Closed-Loop Individualized Curricula (CLIC)**, which we divide into three standardized sub-modules for flexible implementation choices: (1) **AV Evaluation**: At each iteration, we begin by exposing the AV model to a subset of scenarios to assess the current AV capability. (2) **Scenario Selection**: Next, we aim to estimate whether the AV model collides with others in each scenario. To achieve this, we employ a discriminator network trained to predict collision probabilities based on the assessed outcomes, which we term "difficulty predictor". Subsequently, we leverage the predicted labels indicating risk levels to reweight sampling within the scenario library. This provides AV with individualized curricula that align with its current capability. (3) **AV Training**: The AV model is then trained using the scenarios obtained through individualized re-sampling. Overall, CLIC adheres to the principles of continual and curriculum learning by commencing training with simpler samples and progressively introducing more challenging ones, achieving scenario transfer and adaptation out of those diverse yet poorly organized historical scenario libraries for closed-loop AV optimization. The algorithmic design also helps safeguard against catastrophic forgetting [Toneva *et al.*, 2018; Khetarpal *et al.*, 2022] of AV model. Against several competing curriculum-based baselines, experimental results demonstrate that CLIC effectively optimizes AV models to handle more challenging scenarios while minimizing any degradation in performance for easier cases.

2 Related Work

2.1 Scenario-Based Autonomous Driving

Scenario-based autonomous driving has recently become a popular paradigm for training and testing autonomous driving models. Currently, the focus of research in this area mainly lies in scenario generation and tesing AV model, particularly generating extreme scenarios which are rare in NDD by training RL models to control background vehicle (BV). Adaptive Stress Testing [Koren *et al.*, 2018; Koren and Kochenderfer, 2019; Corso *et al.*, 2019; Lee *et al.*, 2020] adopts Monte Carlo tree search and DRL to generate extreme scenarios. Feng et al. [Feng *et al.*, 2020c,a,b; Sun *et al.*, 2021] propose a unified framework for adaptive testing scenario library generation. Bayesian optimization and DRL are employed in different cases. AdvSim [Wang *et al.*, 2021a] and KING [Hanselmann *et al.*, 2022] use a constructed adversarial cost function to explore and generate safety-critical scenarios. $(Re)^2H2O$ [Niu *et al.*, 2023] efficiently generates varied adversarial scenarios by combining NDD with simulation data through hybrid offline-and-online RL.

The scenarios generated from the above methods can be stored by saving the state information of all vehicles at each time step as a static scenario library, which is stable and easy to reproduce. However, according to our research, there is currently a lack of work that utilizes existing large-scale static scenario libraries for training AV models, thus failing to establish an industrial closed-loop of

scenario generation, AV training and testing. Our goal is to complete the missing step in this closed loop, achieving an integrated approach to train and test scenario-based AV models.

2.2 Continual Driving Policy Optimization

Curriculum learning (CL) [Bengio *et al.*, 2009] is a training strategy involving reweighting the training data and designing a series of tasks or examples in increasing difficulty order. By gradually exposing the model to more challenging instances, CL enables a smoother learning process and better generalization performance, prevent the model from becoming trapped in local optima. In addition to its original definition [Bengio *et al.*, 2009], CL has been extended to a series of similar or expanded algorithms, such as self-paced learning (SPL) [Kumar *et al.*, 2010; Jiang *et al.*, 2015; Ren *et al.*, 2018; Klink *et al.*, 2020] and teacher-guided learning [Matiisen *et al.*, 2019; Portelas *et al.*, 2020; Schraner, 2022; Shenfeld *et al.*, 2023]. These approaches have found wide applications in both DL [Graves *et al.*, 2017; Jiang *et al.*, 2018; Hacohen and Weinshall, 2019; Kim and Choi, 2018] and RL [Florensa *et al.*, 2017, 2018; Klink *et al.*, 2021; Cai *et al.*, 2023] domains. In addition, Prioritized Experience Replay (PER) [Schaul *et al.*, 2015] is another method in RL that involves data reweighting. Instead of reweighting each training data, it focuses on reweighting individual transitions, assigning higher priority to transitions that are deemed more important or informative for learning.

The application of CL in AV training is currently not very common, and most of them align with predefined CL [Wang *et al.*, 2021b], and distinguish curricula by discrete factors such as weather and road topology structure [Ozturk *et al.*, 2021], providing future BV trajectories of different lengths [Khaitan and Dolan, 2022], changing the target driving distance [Agarwal *et al.*, 2021] and the quantity of BV [Anzalone *et al.*, 2021, 2022]. In addition, [Qiao *et al.*, 2018] utilize the value function V from RL to measure the learning potential of different tasks, and divide tasks based on the distance range from the starting point to the intersection. Most of the aforementioned works define the curriculum as discrete stages and predefine when to switch between them, which often fall short in terms of automaticity, adaptability and flexibility; and usually train the model with random traffic flow. Although [Khaitan and Dolan, 2022] also employs scenario-based training, it is not applicable to our large-scale scenario library with varying levels of difficulty.

To overcome the aforementioned limitations and better adapt to our objectives and the existing scenario library, we employ the definition of **Data-level Generalized CL** [Wang *et al.*, 2021b]: Curriculum learning refers to the reweighting of the target training set distribution in T training steps. We aim to assign different weights to each scenario in the scenario library automatically, and then perform weighted sampling to obtain training scenarios as individualized curriculum. By continually updating these weights, we can achieve changes in the difficulty distribution of the static scenario library, thus enabling continual optimization of driving policy.

3 Methodology

3.1 Problem Formulation

3.1.1 Scenario Formulation

Our scenarios are based on driving straight on the highway and assume that within our area of interest, there is only one AV ($V^{(0)}$) and N BVs ($V^{(1)}, \ldots, V^{(N)}$). For each vehicle at moment t, we use its lateral and longitudinal position (x, y), concatenated with its velocity v and heading angle θ as the state vector, and the changes in AV's velocity and heading angle between two moments as the action vector: $\mathbf{s}_t^{(i)} = [x_t^{(i)}, y_t^{(i)}, v_t^{(i)}, \theta_t^{(i)}]$, $\mathbf{a}_t^{(0)} = [\Delta v_t^{(0)}, \Delta \theta_t^{(0)}]$. In this way, the complete state and action vectors are $\mathbf{s}_t = [\mathbf{s}_t^{(0)}, \mathbf{s}_t^{(1)}, \ldots, \mathbf{s}_t^{(N)}]^{\mathrm{T}}, \mathbf{a}_t = [\mathbf{a}_t^{(0)}]^{\mathrm{T}}$, while $\mathbf{s}_t^{\mathrm{BV}} = [\mathbf{s}_t^{(1)}, \ldots, \mathbf{s}_t^{(N)}]^{\mathrm{T}}$ only represents the state of all BVs.

A scenario d is defined as a finite sequence of traffic scenes consisting of successive H frames, where each frame contains the state of all BVs, and the initial state of all traffic participants (including the AV and all BVs) is given at the initial time:

$$d = [\mathbf{s}_0, \mathbf{s}_1^{\mathrm{BV}}, \ldots, \mathbf{s}_H^{\mathrm{BV}}] \tag{1}$$

It should be emphasized that this definition of scenarios is AV-agnostic, getting rid of the influence of other AV models with distinct behaviors used to collect those scenarios. Therefore, they can be reapplied to different AV models for training.

Due to the limited precision of the decision modeling at the trajectory level in the SUMO simulator [Behrisch *et al.*, 2011] we use, we choose to manually calculate the kinematic state transitions of AV as follows, while the state transition of BVs is directly provided by the recorded scenario data:

$$
\begin{aligned}
x_{t+\Delta t}^{(0)} &= x_t^{(0)} + v_t^{(0)} \cdot \cos \theta_t^{(0)} \cdot \Delta t \\
y_{t+\Delta t}^{(0)} &= y_t^{(0)} + v_t^{(0)} \cdot \sin \theta_t^{(0)} \cdot \Delta t \\
v_{t+\Delta t}^{(0)} &= v_t^{(0)} + \Delta v_t^{(0)} \\
\theta_{t+\Delta t}^{(0)} &= \theta_t^{(0)} + \Delta \theta_t^{(0)}
\end{aligned}
\tag{2}
$$

3.1.2 Training AV with RL

Consider the development in a traffic environment as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P, \rho, \gamma)$, where P, ρ and γ are the transition probability, initial state distribution and discount factor. $\mathcal{S}$ and $\mathcal{A}$ are state space and action space as outlined above. Reward function $r_t = r(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{R}$ are defined as

$$
r = r_{acc} + r_{vel} + r_{yaw} + r_{lane}
\tag{3}
$$

which includes the following items, and the coefficient values for each item can be found in Table 1:

- *Accident*: Instruct AV to avoid accidents, including collisions with BV and driving off roads:

$$
r_{acc} = -\rho_{acc} \cdot \mathbb{I}_{\{\text{AV has an accident}\}}
\tag{4}
$$

  where $\mathbb{I}$ represents the indicative function, and ρ_{acc} is the corresponding coefficients.

- *Velocity*: Encourage AV to drive faster within the allowed velocity range $[0, v_{max}]$:

$$
r_{vel} = \rho_{vel} \cdot \frac{v_t^{(0)} - v_{max}/2}{v_{max}/2}
\tag{5}
$$

  where $v_t^{(0)}$ is the current velocity of AV, and ρ_{vel} is the corresponding coefficients.

- *Heading direction*: Instruct AV to drive smoothly and drive aligned with a line:

$$
r_{yaw} = -\rho_{yaw} \cdot |\theta|
\tag{6}
$$

  where θ is the heading angle of AV, and ρ_{yaw} is the corresponding coefficients.

- *Lane selection*: Encourage AV to drive on the "best" lane, where the distance between AV and the nearest BV in front of AV on the same lane is the maximum:

$$
r_{lane} = \rho_{lane} \cdot \mathbb{I}_{\{\text{AV on the best lane}\}}
\tag{7}
$$

  where ρ_{lane} is the corresponding coefficients.

Although our scenario library is static, the transition distribution used for training varies as the AV policy changes, because the AV state information is not included in the scenario library but is instead populated through online rollouts to align with the current AV policy. Thus, we choose the online RL algorithm Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018b,a] to solve the MDP problem. The objective function of SAC includes an entropy term $\mathcal{H}$ of the AV policy distribution:

$$
J_\pi(\phi) = \sum_{t=0}^{H} \mathbb{E}_{\mathbf{s}_0 \in \rho, \mathbf{a}_t \sim \pi_\phi(\cdot|\mathbf{s}_t), \mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)} [\gamma^t r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t))]
\tag{8}
$$

where π_ϕ is the AV policy parameterized by ϕ, and α is the temperature hyperparameter in the algorithm. This enables the exploration of more possible actions while maximizing the expected reward discounted by γ, which enhances the robustness of the model and is also more consistent with the application patterns in real-world traffic scenarios.

4

3.2 Algorithmic Framework

We refine our approach into a universal modular framework (Algorithm 1), and provide our specific definitions for each module based on this.

In our universal algorithmic framework, each module is undefined and expandable, which allows us to focus on the relationships between modules and the flow of data, while ignoring the implementation details of each module. It also facilitates the adoption of different specific implementations within our framework in order to propose new methods.

Algorithm 1: Universal Algorithmic Framework

1: **Initialize**: AV policy $\pi_\phi(\mathbf{a}|\mathbf{s})$
2: **Input**: static scenario library $\mathcal{D}$
3: **for** each iteration **do**
4:     info $\leftarrow$ EVALUATEAV($\pi_\phi(\mathbf{a}|\mathbf{s}), \mathcal{D}$)
5:     $d_{\text{train}} \leftarrow$ SELECTSCENARIO($\mathcal{D}$, info)
6:     $\pi_\phi(\mathbf{a}|\mathbf{s}) \leftarrow$ TRAINAV($\pi_\phi(\mathbf{a}|\mathbf{s}), d_{\text{train}}$)
7: **end for**

Our universal algorithmic framework consists of three main stages: AV Evaluation, Scenario Selection, and AV Training. In the **AV Evaluation** stage, we evaluate the current AV model to obtain an estimate of the current capabilities of the AV model, denoted as info in Algorithm 1. The info here can take any form of evaluation information, depending on how the evaluation process and its corresponding output are defined. In the **Scenario Selection** stage, training scenarios are selected from the scenario library based on a certain criterion, in order to explore the boundaries of the AV model's capabilities and achieve the maximum improvement in AV performance. In the **AV Training** stage, any RL algorithm can be used to improve the AV policy in the training scenarios. These three stages are executed sequentially and stop after several iterations.

3.3 Algorithmic Implementation

3.3.1 AV Evaluation

In this stage, we aim to evaluate the capabilities of the current AV model $\pi_\phi(\mathbf{a}|\mathbf{s})$ using a subset of scenarios d_{eval} from the scenario library. Considering that the time cost of this stage should not be too high and the unbiasedness of the scenario distribution should be maintained, we randomly sample m scenarios from the scenario library $\mathcal{D}$ with a total of M scenarios, and sequentially rollout them in the environment to interact with the current AV model. Then we return the results of the interaction, which are a set of labels l_{gt}. Detailed algorithmic steps are shown in Algorithm 2.

Algorithm 2: EVALUATEAV

1: **Initialize**: evaluation scenario number m
2: **Input**: current AV policy $\pi_\phi(\mathbf{a}|\mathbf{s})$, scenario library $\mathcal{D}$
3: $d_{\text{eval}} \leftarrow$ SAMPLE($\mathcal{D}, m$)
4: **for** $d^{(i)}$ in d_{eval} **do**
5:     $\mathbf{s} \leftarrow d_{\mathbf{s}_0}^{(i)}$; done $\leftarrow$ False
6:     **while** not done **do**
7:         $\mathbf{a} \leftarrow \arg\max_{\mathbf{a}} \pi_\phi(\mathbf{a}|\mathbf{s})$
8:         $\mathbf{s}', r,$ done $\leftarrow$ STEP($\mathbf{a}$); $\mathbf{s} \leftarrow \mathbf{s}'$
9:     **end while**
10:    $l_{\text{gt}}^{(i)} \leftarrow \mathbb{I}_{\{\text{AV colides with BV}\}}$
11: **end for**
12: **return** l_{gt}

3.3.2 Scenario Selection

In order to continually optimize driving policy and enhance the ability of AV to handle extreme scenarios, it is important to focus training on more challenging scenarios while including a small number of less difficult scenarios to prevent forgetting. Thus, associating the difficulty of each scenario with its sampling weight is an intuitive approach. However, currently there is no perfect benchmark to assess scenario difficulty, meanwhile scenario difficulty is also an individualized metric. Thus we make use of the results from each evaluation stage as the information for the current AV model.

We treat the scenarios used for evaluation d_{eval} and their collision labels l_{gt} as the ground truth, and then employ a supervised learning approach to train a difficulty predictor model $p_\psi(d|\phi)$, where d can be any scenario that meets the definition in Equation 1. We utilize a three-layer feedforward neural network [Goodfellow *et al.*, 2016] parameterized by ψ as the difficulty predictor $p_\psi(\cdot|\phi)$, and flatten the entire scenario data d into a one-dimensional vector as the input of the predictor. Binary cross-entropy loss (BCELoss) is used as the loss function:

$$\mathcal{L}_{\mathrm{pred}}(\psi) = -\mathbb{E}_{d^{(i)} \sim d_{\mathrm{eval}}}\left[l_{\mathrm{gt}}^{(i)} \cdot \log(l_{\mathrm{pred}}^{(i)}) + (1 - l_{\mathrm{gt}}^{(i)}) \cdot \log(1 - l_{\mathrm{pred}}^{(i)})\right] \tag{9}$$

where $l_{\mathrm{pred}}^{(i)} = p_\psi(d^{(i)}|\phi)$ is the predicted label value of the i-th scenario $d^{(i)}$. The number of training epochs is determined through experiments to ensure that the predictor neither underfits nor overfits the training data. The predictor is then applied to the entire scenario library $\mathcal{D}$ to obtain the predicted label values l_{all} for each scenario. These predicted label values are then used as weights for weighted sampling, resulting in a batch of training scenarios d_{train}. Specifically, the probability of sampling the i-th scenario $d^{(i)}$ is:

$$P(d^{(i)} \in d_{\mathrm{train}}) = \frac{l_{\mathrm{all}}^{(i)}}{\sum_{i=1}^{M} l_{\mathrm{all}}^{(i)}} = \frac{p_\psi(d^{(i)}|\phi)}{\sum_{i=1}^{M} p_\psi(d^{(i)}|\phi)} \tag{10}$$

Algorithm 2 and Algorithm 3 together form the process of individualized curriculum design.

Algorithm 3: SELECTSCENARIO

1: **Initialize**: difficulty predictor $p_\psi(\cdot|\phi)$, learning rate α, training scenario number n
2: **Input**: scenario library $\mathcal{D}$, evaluation scenarios d_{eval}, labels of evaluation scenarios l_{gt}
3: **for** each epoch **do**
4:     $l_{\mathrm{pred}} \leftarrow p_\psi(d_{\mathrm{eval}}|\phi)$
5:     $\psi \leftarrow \psi - \alpha\nabla_\psi\mathcal{L}_{\mathrm{pred}}(\psi)$ ▷ Equation 9
6: **end for**
7: $l_{\mathrm{all}} \leftarrow p_\psi(\mathcal{D}|\phi)$
8: $d_{\mathrm{train}} \leftarrow \text{WEIGHTEDSAMPLE}(\mathcal{D}, n, l_{\mathrm{all}})$ ▷ Equation 10
9: **return** d_{train}

3.3.3 AV Training

In this stage, we employ online RL algorithm SAC to train the AV model. At each epoch, we first shuffle the order of the training scenarios and then rollout each scenario one by one, updating the parameters of the AV model based on SAC.
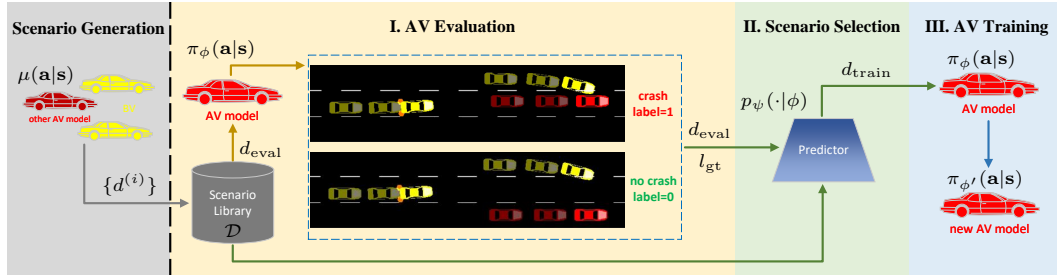

Figure 1: Overall Algorithmic Architecture

By incorporating the specific implementation of the aforementioned module into the universal algorithmic framework, we obtain an overall architecture illustrated in Figure 1, which includes the three modules mentioned above. The scenario generation stage for other AV model $\mu(\mathbf{a}|\mathbf{s})$ on the left of the figure has already been extensively studied and is therefore outside the scope of this algorithm. In each training iteration, a batch of scenarios d_{eval} is randomly sampled from the static scenario library $\mathcal{D}$ to evaluate the current AV model. These scenarios are rolled out in the environment and interacts with AV to obtain collision labels l_{gt}. These scenarios and labels are then used to train the difficulty predictor $p_\psi(\cdot|\phi)$, which, after training, predicts labels for all scenarios in $\mathcal{D}$. These predicted labels l_{all} are used as weights to perform weighted sampling on all scenarios, resulting in the training scenarios d_{train}. Finally, the AV model $\pi_\phi(\mathbf{a}|\mathbf{s})$ is trained using an online RL algorithm on these training scenarios, completing a full training iteration.

4 Experiments

We provide evidence for the superiority of CLIC in training AV models by addressing the following questions:

- Can CLIC train safer AV models than other baselines? (Sec. 4.2.1)

- Is CLIC strengthening the AV model while increasing the difficulty of training scenarios selected by the predictor throughout the training process? (Sec. 4.2.2)

- Can CLIC select training scenarios specifically tailored to the defects of a particular vehicle? (Sec. 4.2.3)

- How does CLIC reweight the data distribution of the scenario library according to AV capability? (Sec. 4.2.4)

4.1 Experiment Settings

4.1.1 Dataset

All traffic scenarios used in our experiments are generated by $(Re)^2H2O$ [Niu *et al.*, 2023]. Refer to Appendix A for more details and visualization results about the dataset.

4.1.2 Baselines

We select five baselines for comparison:

- ***SAC w/ rand***: Simply sample a batch of training scenarios randomly from the scenario library.

- ***SAC w/ rand+fail***: In each training iteration, half of the training scenarios are randomly sampled from the scenario library, while another half consists of scenarios that failed in the corresponding AV Evaluation stage.

- ***SAC w/ fail***: All training scenarios are chosen from those that fail in AV Evaluation stage.

- ***PER***: Apply PER [Schaul *et al.*, 2015] into RL based on *SAC w/ rand*.

- ***PCL***: Following the idea of predefined curriculum learning (PCL) [Wang *et al.*, 2021b], we divide the training process into four phases of the curriculum based on the number of BV (1~4). The training set for each phase adds scenarios with a higher number of BV based on the previous phase.

4.1.3 Evaluation Metrics

To evaluate the capabilities and training effectiveness of the AV model in extreme scenarios, we employed the following metrics[1]:

- **SR** (%): Success rate, referring to the success rate of the AV model in all scenarios after the final iteration of training. It reflects the overall safety performance of the AV model in various extreme scenarios.

- **Confusion Matrix**: To distinguish between the difficulty levels of different scenarios and to highlight the effectiveness of the AV model in high-difficulty scenarios, we also borrow the concept of confusion matrix in supervised learning: we define the label of successful scenarios as 0, and that of failed scenarios as 1. The test results before training are treated as ground truth, while those after training are treated as predicted values. Specifically, we compute the following two metrics:

  - **FNR** (%) $= \frac{FN}{TP+FN}$, which represents the proportion of successful scenarios after training that are failed before.

  - **TNR** (%) $= \frac{TN}{TN+FP}$, which represents the proportion of successful scenarios after training that are successful before.

- **CPS** (s^{-1}) and **CPM** (m^{-1}): As defined in [Niu *et al.*, 2023], CPS is Average Collision Frequency Per Second and CPM is Average Collision Frequency Per Meter, calculated as following:

$$\text{CPS} = \frac{N_{col}}{T_{total}}, \quad \text{CPM} = \frac{N_{col}}{D_{total}} \tag{11}$$

---

[1]We expect higher values for SR, FNR and TNR, and lower values for CPS and CPM.
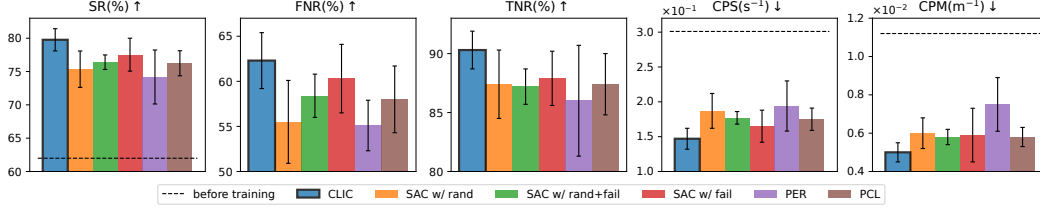
Figure 2: Comparison of various metrics for CLIC and baselines.

where N_{col} is the collision number of AV, T_{total} and D_{total} are the total testing time and the total driving distance traveled by AV along the road direction.

4.2 Experimental Results

4.2.1 Comparison Experiment

We conduct tests on all scenarios in the scenario library to compare different baselines on the aforementioned metrics, and present the results from 5 random seeds as shown in Figure 2. CLIC outperforms all baselines in all metrics, achieving the highest SR, FNR, TNR and the lowest CPS and CPM. This indicates that our training method not only enables the AV model to learn more challenging scenarios it couldn't handle before but also minimizes the risk of forgetting previously learned scenarios to the utmost. Although *SAC w/ fail* focuses on training failure scenarios and achieves relatively good results on FNR, it is also easy to forget simple scenarios, leading to a little bad performance on TNR. Furthermore, *SAC w/ rand* and *PER* perform the worst, further illustrating the importance of a well-designed curriculum compared to blindly training on randomly sampled scenarios from the scenario library. We also included the SR, CPS, and CPM of the AV model with random initialization before training in the figure. It can be observed that even for an untrained AV model, its SR is already quite high (62%). Therefore, using random sampling of scenarios for training like *SAC w/ rand* and *PER* would inevitably lead to inefficient training, wasting time on scenarios that the AV model does not require training on.

4.2.2 Matrix Experiment

In addition to testing the performance change of the AV model before and after training, we also aim to verify the changes in the capabilities of the AV model and the difficulty of scenarios selected by the predictor throughout the training process. Therefore, we conduct the following matrix experiments: we save the AV model and predictor model at each training iteration, and sequentially use each predictor to select scenarios and test them with each AV model. This process yield a $T \times T$ matrix of SR, where T represents the total number of training iterations. For instance, the element "*AV iteration* = 3, *predictor iteration* = 4" represents the testing SR of the AV model trained after three iterations when tested on the scenarios selected by the predictor in the fourth iteration. As shown in Figure 3, the experimental results demonstrate the following characteristics:

- The AV models with a higher number of training iterations achieved higher SR among the scenarios selected by each specific predictor, indicating that the capability of the AV model is indeed continually increasing.

- For each specific AV model, there is a decreasing trend in SR among the scenarios selected by subsequent predictors, indicating that the difficulty of the scenarios selected by the predictor is indeed increasing.

- For the AV model in the first training iteration, it exhibits a higher SR on scenarios selected by some of subsequent predictors. This is primarily due to the lower capability of the AV model at this stage, rendering the difficulty level of predictions by the predictors insignificant. This observation further emphasizes the importance of individualized curriculum design in CLIC.

4.2.3 Analyses on Individualization

To demonstrate that CLIC can indeed generate individualized curricula that identify specific defects in AV model and select more targeted training scenarios accordingly, we intentionally disable a trained AV model to get less aware of BVs on the left front side. We then let this disabled AV go through CLIC pipeline and compare the selected scenarios with the ones selected for the AV model without defects. We analyze the distribution of distances between the AV and the BVs on its left front side in these scenarios illustrated in Figure 4. As expected, these BVs are positioned
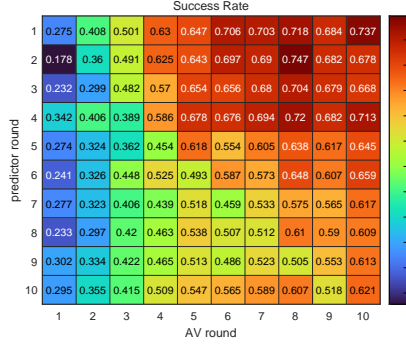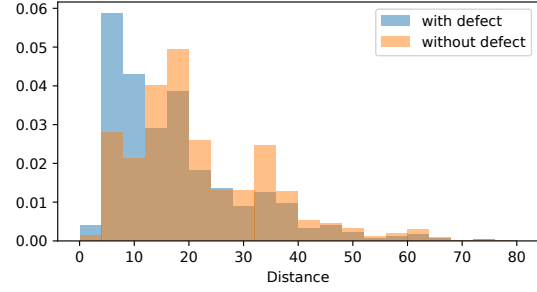
Figure 3: Results of the matrix experiment.

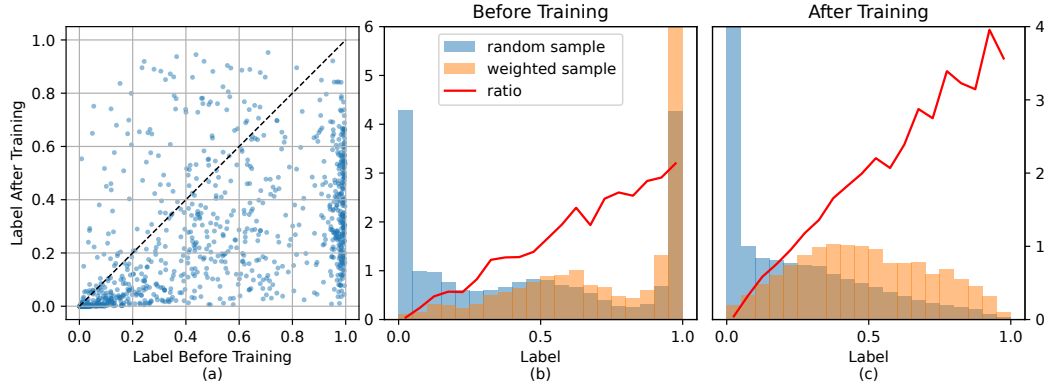Figure 4: Distribution of distances between the AV and the BVs on its left front side.

Figure 5: **(a)** Changes of partial individual scenario labels before and after training. **(b)(c)** Label distribution before and after training.

closer to the AV, increasing the risk of collisions. Furthermore, CLIC selected a significantly higher proportion (**27.87%**) of scenarios for AVs with defect where BVs are positioned on the left front side, surpassing the selected proportion (**17.19%**) of the AV model with no defect by a substantial margin. These clearly echo our intuition of providing individualized curricula that target specific defects in training AV model.

4.2.4 Analyses on Scenario Reweighting

To showcase the changes in the distribution of scenario weights before and after training, we tracked the changes in predicted label values for individual scenarios before and after training. These changes are presented as a scatter plot (Figure 5(a)). The scenarios in the figure are concentrated in two regions: first, scenarios with initially small label values tend to be concentrated around 0 after training; second, scenarios with label values close to 1 before training show varying degrees of decrease in label values after training. Additionally, the majority of scenarios fall below the diagonal line $y = x$, indicating that these scenarios indeed become simpler for the AV model after training. This observation further confirms the conclusion drawn in 4.2.2.

Moreover, to further illustrate the overall scenario reweighting, we also plot histograms of the label distributions obtained through random sampling and weighted sampling before and after training, as shown in Figure 5(b)(c). It can be observed that before training, the predicted labels are concentrated at both ends of the interval, while the rest of the distribution appears relatively uniform. However, with weighted sampling, the labels are concentrated near 1, indicating the selection of harder scenarios for the current AV model. After training, as the AV model becomes sufficiently powerful, the predicted labels near 1 significantly decrease, and more labels concentrate near 0. Nonetheless, in order to select relatively hard training scenarios, weighted sampling still acquires a considerable number of scenarios with larger labels. To provide a more intuitive demonstration of the weighted sampling, we also plot the ratio curve of the normalized frequencies corresponding to the two sampling methods. As expected, the ratio curve maintains a proportional relationship for each sampling instance.

5   Conclusion and Future Work

In this paper, we develop a scenario-based continual driving policy optimization framework with Closed-Loop Individualized Curricula (CLIC) technique, composed of three sub-modules for flexible implementation choices: AV Evaluation, Scenario Selection and AV Training. CLIC approaches AV Evaluation as a difficulty prediction task by training a discriminator on AV collision labels to estimate the potential failure probability of AV within corresponding scenarios. With the prediction results from the discriminator, CLIC reweights scenarios to select individualized curricula for AV training that incorporate less easy cases with more challenging ones according to current AV capability. To summarize, CLIC not only fully exploits the vast historical scenario library for closed-loop AV training rather than just for AV testing, but also facilitates AV improvement by individualizing its training on more helpful scenarios that match its current capability out of the diverse yet poorly organized library. Through extensive experimental analyses, CLIC outperforms other competing curriculum-based baselines, especially in handling safety-critical scenarios, while minimizing the degradation of performance in regular scenarios. For future work, we are interested in exploring scenario libraries with more complex road topologies and other types of traffic participants, as well as investigating theoretical guarantees for AV improvement within these individualized curricula. We also believe that CLIC is not only applicable to the field of autonomous driving policy optimization, but also a general method for complex tasks and continual learning in robotics.

References

Pranav Agarwal, Pierre De Beaucorps, and Raoul De Charette. Sparse curriculum reinforcement learning for end-to-end driving. *arXiv preprint arXiv:2103.09189*, 2021. 3

Luca Anzalone, Silvio Barra, and Michele Nappi. Reinforced curriculum learning for autonomous driving in carla. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3318–3322. IEEE, 2021. 3

Luca Anzalone, Paola Barra, Silvio Barra, Aniello Castiglione, and Michele Nappi. An end-to-end curriculum learning approach for autonomous driving scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19817–19826, 2022. 3

Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011. 4

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009. 3

Yuanying Cai, Chuheng Zhang, Hanye Zhao, Li Zhao, and Jiang Bian. Curriculum offline reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 1221–1229, 2023. 3

Zhong Cao, Shaobing Xu, Huei Peng, Diange Yang, and Robert Zidek. Confidence-aware reinforcement learning for self-driving cars. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7419–7430, 2021. 2

Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019. 2

Wenhao Ding, Baiming Chen, Minjun Xu, and Ding Zhao. Learning to collide: An adaptive safety-critical scenarios generating method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2250. IEEE, 2020. 2

Shuo Feng, Yiheng Feng, Haowei Sun, Shan Bao, Yi Zhang, and Henry X Liu. Testing scenario library generation for connected and automated vehicles, part ii: Case studies. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5635–5647, 2020. 2

Shuo Feng, Yiheng Feng, Haowei Sun, Yi Zhang, and Henry X Liu. Testing scenario library generation for connected and automated vehicles: An adaptive framework. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1213–1222, 2020. 2

Shuo Feng, Yiheng Feng, Chunhui Yu, Yi Zhang, and Henry X Liu. Testing scenario library generation for connected and automated vehicles, part i: Methodology. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1573–1582, 2020. 2

Shuo Feng, Haowei Sun, Xintao Yan, Haojie Zhu, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953):620–627, 2023. 2

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017. 3

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018. 3

Daniel J Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020. 2

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 6

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. Pmlr, 2017. 3

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. 4

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 4, 16

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International conference on machine learning*, pages 2535–2544. PMLR, 2019. 3

Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *European Conference on Computer Vision*, pages 335–352. Springer, 2022. 2

Zhiyuan Huang, Henry Lam, David J LeBlanc, and Ding Zhao. Accelerated evaluation of automated vehicles using piecewise mixture models. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2845–2855, 2017. 2

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 3

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018. 3

Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016. 1

Shivesh Khaitan and John M Dolan. State dropout-based curriculum reinforcement learning for self-driving at unsignalized intersections. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12219–12224. IEEE, 2022. 3

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022. 2

Tae-Hoon Kim and Jonghyun Choi. Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904*, 2018. 3

Pascal Klink, Carlo D'Eramo, Jan R Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9216–9227, 2020. 3

Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Boosted curriculum reinforcement learning. In *International Conference on Learning Representations*, 2021. 3

Mark Koren and Mykel J Kochenderfer. Efficient autonomy validation in simulation with adaptive stress testing. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4178–4183. IEEE, 2019. 2

Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018. 2

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010. 3

Ritchie Lee, Ole J Mengshoel, Anshu Saksena, Ryan W Gardner, Daniel Genin, Joshua Silbermann, Michael Owen, and Mykel J Kochenderfer. Adaptive stress testing: Finding likely failure events with reinforcement learning. *Journal of Artificial Intelligence Research*, 69:1165–1201, 2020. 2

Xiaoyi Li. A scenario-based development framework for autonomous driving. *arXiv preprint arXiv:2011.01439*, 2020. 2

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019. 3

Demin Nalic, Tomislav Mihalj, Maximilian Bäumler, Matthias Lehmann, Arno Eichberger, and Stefan Bernsteiner. Scenario based testing of automated driving systems: A literature survey. In *FISITA web Congress*, volume 10, 2020. 2

Haoyi Niu, Jianming Hu, Zheyu Cui, and Yi Zhang. Dr2l: Surfacing corner cases to robustify autonomous driving via domain randomization reinforcement learning. In *Proceedings of the 5th International Conference on Computer Science and Application Engineering*, pages 1–8, 2021. 2

Haoyi Niu, Kun Ren, Yizhou Xu, Ziyuan Yang, Yichen Lin, Yi Zhang, and Jianming Hu. (re)2h2o: Autonomous driving scenario generation via reversely regularized hybrid offline-and-online reinforcement learning. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–8, 2023. 2, 7, 14

Anil Ozturk, Mustafa Burak Gunel, Resul Dagdanov, Mira Ekim Vural, Ferhat Yurdakul, Melih Dal, and Nazim Kemal Ure. Investigating value of curriculum reinforcement learning in autonomous driving under diverse road and weather conditions. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 358–363. IEEE, 2021. 3

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020. 3

Zhiqian Qiao, Katharina Muelling, John M Dolan, Praveen Palanisamy, and Priyantha Mudalige. Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1233–1238. IEEE, 2018. 3

Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17305–17315, 2022. 2

Zhipeng Ren, Daoyi Dong, Huaxiong Li, and Chunlin Chen. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, 29(6):2216–2226, 2018. 3

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. 3, 7

Yanick Schraner. Teacher-student curriculum learning for reinforcement learning. *arXiv preprint arXiv:2210.17368*, 2022. 3

Idan Shenfeld, Zhang-Wei Hong, Aviv Tamar, and Pulkit Agrawal. Tgrl: Teacher guided reinforcement learning algorithm for pomdps. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023. 3

Haowei Sun, Shuo Feng, Xintao Yan, and Henry X Liu. Corner case generation and analysis for safety assessment of autonomous vehicles. *Transportation research record*, 2675(11):587–600, 2021. 2

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018. 2

Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021. 1, 2

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2021. 3, 7

Jingxuan Yang, Haowei Sun, Honglin He, Yi Zhang, Henry X Liu, and Shuo Feng. Adaptive safety evaluation for connected and automated vehicles with sparse control variates. *IEEE Transactions on Intelligent Transportation Systems*, 2023. 2

Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1

Ding Zhao, Henry Lam, Huei Peng, Shan Bao, David J LeBlanc, Kazutoshi Nobukawa, and Christopher S Pan. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. *IEEE transactions on intelligent transportation systems*, 18(3):595–607, 2016. 2

Ding Zhao, Xianan Huang, Huei Peng, Henry Lam, and David J LeBlanc. Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):733–744, 2017. 2

Ziyuan Zhong, Yun Tang, Yuan Zhou, Vania de Oliveira Neves, Yang Liu, and Baishakhi Ray. A survey on scenario-based testing for automated driving systems in high-fidelity simulation. *arXiv preprint arXiv:2112.00964*, 2021. 2

Appendix

A Dataset Analysis and Visualization

All of the scenario data we use is generated by $(\text{Re})^2\text{H2O}$ [Niu *et al.*, 2023] and saved in the form of static vectors after preprocessing according to the definition in Section 3.1.1, making it convenient to flexibly load into the data stream for reproduction when needed. This scenario library records 65494 extreme scenarios (excluding scenarios where BVs collided with each other already) formed when 1~4 BVs attempted to attack AV on a 3-lane straight highway with a length of about 200m, with a sampling interval of $\Delta t = 0.04\text{s}$. Among them, 53040 scenarios showed successful collisions between BV and AV, with a collision rate of 81.0%. Figure 6 shows an illustrated example of the scenarios where one BV collides with AV.

Figure 6: An illustrated example of the scenarios.

During scenario generation, certain limitations were imposed on the actions of BVs to meet common vehicle dynamics constraints. The acceleration of BVs is limited to $[-7.84, 5.88](\text{m/s}^2)$ (i.e. $[-0.8g, 0.6g]$), while the angular velocity is limited to $[-\pi/3, \pi/3](\text{rad/s})$. In addition, the maximum velocity limit on the road is 40m/s and negative velocity (i.e. reversing) is not allowed.

In order to better illustrate some of the features of the scenario library, we have calculated the distribution of the following features and visualized them: the yaw, velocity and acceleration of BVs, the distance between BVs, the initial distance between BVs and AV, and the initial position of BV relative to AV. Figure 7 $\sim$ 9 show these features respectively.
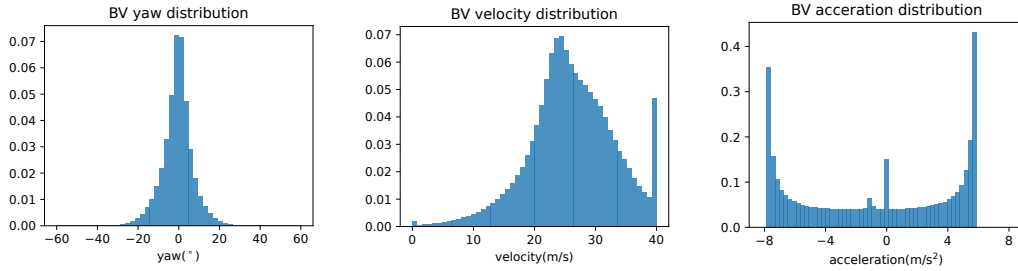

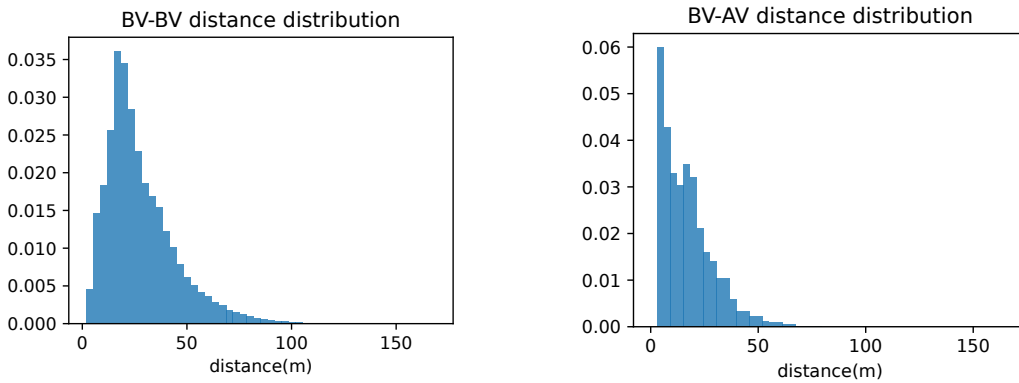Figure 7: The yaw, velocity and acceleration distribution of all BVs.

Figure 8: **Left**: The distance distribution between BVs. **Right**: The initial distance distribution between BVs and AV.
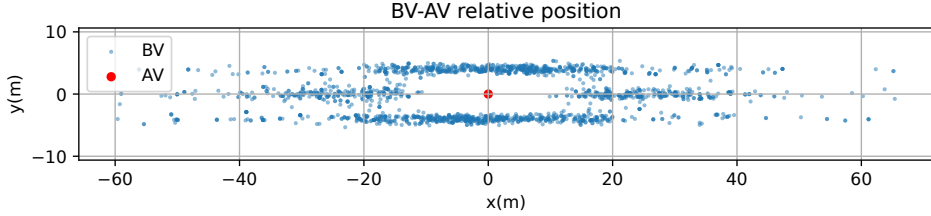
Figure 9: The initial position of partial BV relative to AV.

From the above feature distribution statistics, it can be seen that although the scenario library contains a large number of scenarios where AV accidents occur during generation, there are also many scenarios where BVs' driving behavior patterns are quite conventional, and the BVs are relatively sparse. Therefore, it is highly likely that these scenarios overfitted the AV model at the time of generation and are not applicable to other AV models, resulting in high SR in our testing experiment. In addition, these feature distributions also reflect the diversity of the scenario library data, making them more suitable for our algorithmic framework that utilizes closed-loop feedback for individualized training scenario selection.

B Implementation Details

Due to the fact that neural networks must have fixed dimensional inputs, but the different number of BVs and timesteps in the scenarios can result in inconsistent input dimensions, we have to initialize the neural network by obtaining the maximum possible input dimension in advance, and fill in the remaining dimensions with zeros when the input dimension is insufficient. In this way, the actual input dimension of the predictor is $6 \times (1 + 4 \times 100) = 2406$ (Each vehicle's data at each moment contains 6 dimensions, including timestep t, vehicle ID, $x_t^{(i)}$, $y_t^{(i)}$, $v_t^{(i)}$, $\theta_t^{(i)}$; all scenario data contain up to 4 BVs, with a maximum timestep of 100; 1 represents the AV information at the initial moment), while the actual input state space dimension of the AV model is $4 \times (1 + 4) = 20$ (The state of each vehicle includes 4 dimensions: $x_t^{(i)}$, $y_t^{(i)}$, $v_t^{(i)}$, $\theta_t^{(i)}$, with a total of 1 AV and no more than 4 BVs). On the other hand, the object of action space is only the AV itself, so its dimension is determined to be 2 ($\Delta v_t^{(i)}$, $\Delta \theta_t^{(i)}$) according to Section 3.1.1.

Table 1 lists the values of the coefficients corresponding to each reward function item, which have been adjusted through certain experiments.

Table 1: Coefficients of the reward function

Coefficient	Description	Value
ρ_{acc}	Coefficient of the accident reward items	40
ρ_{vel}	Coefficient of the velocity reward items	0.8
ρ_{yaw}	Coefficient of the yaw reward items	$6/\pi$
ρ_{lane}	Coefficient of the lane reward items	2.0

The network structure and hyperparameters of our algorithm are listed in Table 2. Except for the temperature parameter $\alpha = 0.1$ which is adjusted through experiments, most of these hyperparameters are based on empirical values.

All of our experiments are conducted on a Linux server equipped with an 8-core Intel(R) Xeon(R) CPU E5-2620 v4 and 3 NVIDIA Geforce GTX 1080Ti chips. Under the above experimental setup, the entire closed-loop training process takes about 6 hours, with the *AV training* stage in Section 3.3.3 accounting for an average of over **83.0%** of the time. This indicates that our individualized curriculum generation process does not bring significant time costs and computational burden.

C Additional Experiment Results

C.1 Additional Ablations, Baselines and Metrics

In addition to the baselines mentioned in Section 4.1.2, our experiment also includes the following experimental settings for comparison:

Table 2: Hyperparameters

Hyperparameter	Description	Value
Evaluate size	Number of scenarios tested in AV Evaluation stage	4096
Train size	Number of scenarios trained in AV Training stage	128
Layer number	Number of hidden layers in each network (all)	3
Hidden size	Number of hidden units in each hidden layer	256
Batch size	Number of examples in each batch when training the predictor	128
Learning rate	The step size in network gradient descent (all)	10^{-4}
Optimizer	The optimizer used in gradient descent (all)	Adam
Nonlinearity	Nonlinear activation functions in networks	ReLU
Nonlinearity (predictor output)	Nonlinear activation function in the output layer of the predictor	Softmax
Nonlinearity (policy output)	Nonlinear activation function in the output layer of the policy net	tanh
Iterations T	The number of iterations of the entire closed-loop algorithm	10
Epochs	The number of training epochs when training the predictor	20
Episodes	The number of training episodes on each scenario	10
Temperature parameter α	Entropy regularization coefficient in SAC objective function	0.1
Buffer size	The maximum capacity of the replay buffer in SAC	10^6
Discount factor γ	The future reward discount factor in RL	0.99
Target smoothing coefficient τ	Update coefficient for target network parameter soft update	0.01

- ***alpha_0.2*** and ***alpha_0.15***: In Section B, we set the temperature parameter α as 0.1, and in fact, we also try $\alpha = 0.15$ and $\alpha = 0.2$ for the experiment and compare the results. The larger the value of α, the more inclined the model is to explore policy diversity.

- ***auto_alpha***: As shown in [Haarnoja *et al.*, 2018b], the temperature parameter α can also be designed with a loss function and automatically adjusted by gradient decent:

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t}[-\alpha \log \pi_t(\mathbf{a}_t|\mathbf{s}_t) - \alpha\bar{\mathcal{H}}]$$
$$\alpha \leftarrow \alpha - \lambda\hat{\nabla}_\alpha J(\alpha) \tag{12}$$

where $\bar{\mathcal{H}}$ is the target entropy, and λ is the learning rate. Here $\bar{\mathcal{H}}$ is set as 0, and λ remains consistent with others, i.e. 10^{-4}.

- ***dropout***: Add dropout process during predictor training to prevent overfitting, and set the dropout probability as 0.2.

- ***batch_1:1***: In order to prevent the imbalance of positive and negative examples during the predictor training process from leading to poor training results, the number of positive and negative examples in each batch is ensured to be 1:1 by copying examples with the fewer labels.

- ***order***: The algorithm settings are basically the same as CLIC, but when selecting training scenarios, they are not weighted sampled based on the predicted difficulty labels. Instead, the scenarios are sorted in order according to their labels, and then all scenarios are divided into intervals according to the order. From each interval we randomly sample one scenario to form the training scenarios.

- ***PCL_label***: In addition to using the number of BV as a difficulty classification basis for PCL, the initial predicted difficulty labels can also be divided into different intervals for PCL, called *PCL_label*, and correspondingly, the baseline *PCL* in Section 4.1.2 is called *PCL_bv_num* in Table 3. Here, all scenarios are divided into 5 curriculum stages at intervals of 0.2 (The range of label is $[0, 1]$), with 2 iterations of training for each stage.

In addition to the safety performance metrics mentioned in Section 4.1.3, we also considered more metrics including efficiency and comfort:

- **vel** (m/s): The average velocity of AV tested in all scenarios.

- **succ_vel** (m/s): The average velocity of AV tested in successful scenarios, reflecting the efficiency performance of AV while ensuring safety.

- **acc** (m/s^2): The average acceleration of AV tested in all scenarios.

- **jerk** (m/s^3): The average derivative of acceleration over time of AV tested in all scenarios.

- **ang_vel** (rad/s): The average angular velocity of AV tested in all scenarios.

- **lat_acc** (m/s^2): The average lateral acceration of AV tested in all scenarios.

Among them, the first two metrics reflect the efficiency of AV and are expected to be larger, while the last four metrics reflect the comfort of AV and are expected to be smaller.

C.2 Complete Results of Comparative Experiments

Adding the above experimental conditions or baselines and metrics, the complete results of the comparative experiments are shown in Table 3.

The experimental results indicate that the CLIC method (including fine-tuning α) has achieved the best performance in various safety and efficiency related metrics. When $\alpha = 0.15$, the efficiency of AV is the highest, and when $\alpha = 0.2$, AV performs quite well in terms of comfort. In addition, thanks to the velocity reward item in our reward function, the AV models trained by various methods have made gratifying progress in efficiency compared to untrained AV model. However, improvements in the training process of the predictor are not reflected in the final results, and the loss function of the predictor itself also dose not significantly decrease further after these improvements. The method of selecting training scenarios in order of difficulty labels is indeed not as good as weighted sampling. Furthermore, as mentioned in Section 4.2.1, PCL has not fully utilized the powerful advantages of curriculum learning, as its predefined curriculum settings completely ignore the ability feedback of the current AV model, thus lacking adaptability.

It is also worth mentioning that although the untrained AV model has shown seemingly good SR, they have exposed their practical shortcomings in CPS and CPM, more than twice as high as CLIC. Through comfort-related metrics, it can also be seen that the untrained AV model is difficult to take action to avoid accidents and only engages in trivial driving in a naïve manner. This also reflects the importance of individualized curricula, otherwise these once challenging scenarios would lose their transferability and the value of reuse even in such a naïve model.

D Qualitative Results

Through SUMO GUI, we visualize some experimental results and obtain these qualitative results to support our conclusion.
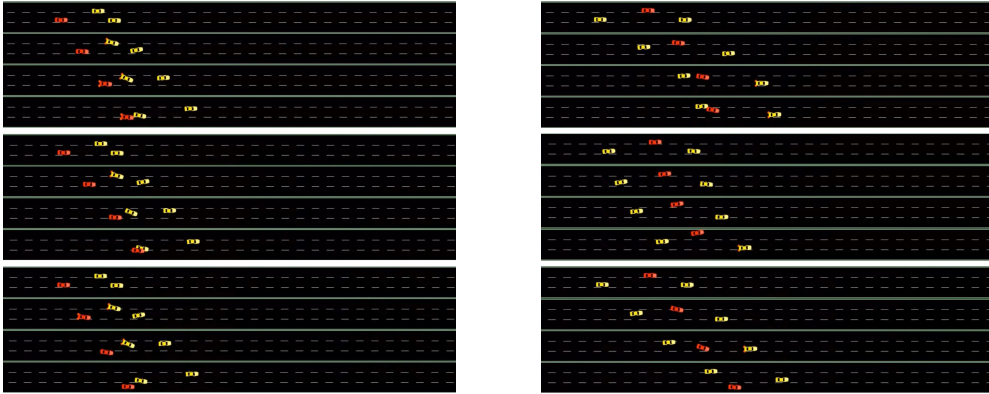
D.1 Comparative Experiment

Figure 10: The performance of different AV models in two scenarios. From top to bottom, they are: **untrained**, **trained by *SAC w/ rand*** (no curriculum), and **trained by CLIC**. Only the AV model trained by CLIC can successfully handle these two scenarios and avoid accidents.
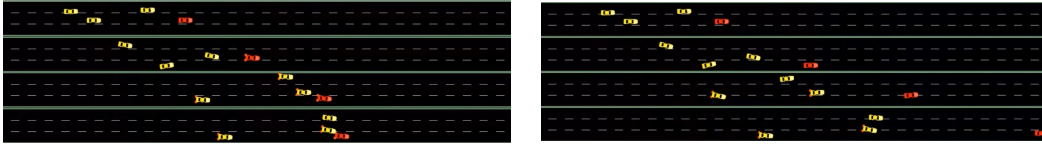
D.2 Matrix Experiment

Figure 11: A result in the matrix experiment with fixed *predictor iteration* $= 5$. According to the predicted labels of the fixed predictor, weighted sampling is performed to obtain the same test scenario. In the **left** image, *AV iteration* $= 1$, and the AV has experienced an accident; In the **right** image, *AV iteration* $= 10$, and the AV has successfully passed the test.
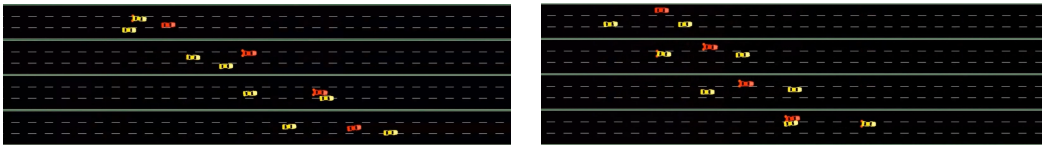

Figure 12: A result in the matrix experiment with fixed *AV iteration* $= 5$. In the **left** image, *predictor iteration* $= 1$, and the selected scenario is relatively simple, where AV successfully passed the test; In the **right** image, *predictor iteration* $= 10$, the selected scenario is quite difficult, where the AV experienced an accident.

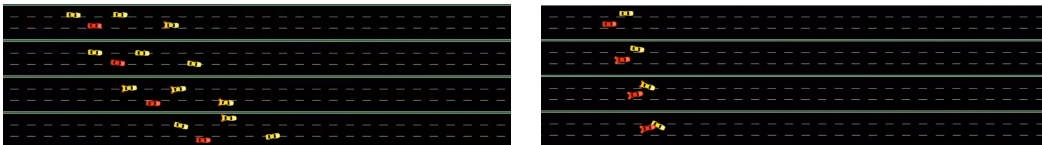## D.3 Analyses on Individualization

Figure 13: An individualized scenario selection result in Section 4.2.3. On the **left** is one scenario selected for normal AV models, and on the **right** is one scenario selected for AV models with left front perception defects. It can be observed that in the scenario on the right, the BV in left front of the AV is significantly closer and more aggressive.

Table 3: Complete results of comparetive experiments

	SR↑	FNR↑	TNR↑	CPS↓	CPM↓	vel↑	succ_vel↑	acc↓	jerk↓	ang_vel↓	lat_acc↓
untrained	61.99±4.03	—	—	0.301±0.044	0.0112±0.0017	27.01±0.31	26.88±0.72	0.61±0.19	0.18±0.05	0.028±0.013	0.76±0.36
CLIC	**79.76±1.67**	**62.3±3.1**	**90.3±1.6**	**0.147±0.015**	0.0050±0.0005	28.92±0.92	29.32±1.05	4.06±0.24	5.70±0.51	0.229±0.015	6.41±0.47
(alpha_0.2)	78.84±0.97	61.4±2.9	89.2±1.6	0.148±0.024	**0.0049±0.0009**	30.18±0.86	30.78±1.09	**3.55±0.14**	3.68±0.40	0.188±0.013	5.37±0.35
(alpha_0.15)	76.45±1.68	60.8±2.8	85.8±3.7	0.177±0.015	0.0055±0.0004	**31.96±0.98**	**32.67±1.02**	4.15±0.43	3.87±1.17	0.206±0.009	6.16±0.33
(auto_alpha)	77.66±2.58	59.9±4.9	88.1±3.1	0.165±0.024	0.0053±0.0007	31.13±0.53	31.57±0.76	3.92±0.08	4.61±0.38	0.201±0.008	5.88±0.27
(dropout)	76.52±3.05	58.7±5.6	87.5±1.7	0.179±0.027	0.0060±0.0014	30.50±2.11	30.82±2.40	4.38±0.30	4.77±1.33	0.222±0.014	6.32±0.34
(batch_1:1)	78.34±3.22	63.0±4.9	87.6±1.6	0.160±0.027	0.0055±0.0010	29.33±1.84	30.04±2.08	4.28±0.32	5.09±0.67	0.225±0.008	6.32±0.45
(order)	76.07±1.63	57.0±2.9	87.6±2.8	0.175±0.022	0.0058±0.0006	30.48±1.74	31.37±1.93	4.28±0.25	4.32±0.35	0.219±0.017	6.21±0.67
SAC w/ rand	75.35±2.73	55.5±4.6	87.4±2.9	0.187±0.025	0.0060±0.0008	31.42±1.11	31.96±1.30	4.12±0.29	3.74±0.56	0.202±0.011	5.91±0.43
SAC w/ rand+fail	76.40±1.09	58.4±2.4	87.2±1.5	0.177±0.009	0.0058±0.0004	30.78±1.77	31.33±2.06	3.97±0.17	4.37±0.61	0.203±0.016	5.94±0.50
SAC w/ fail	77.53±2.47	60.3±3.8	87.9±2.3	0.165±0.023	0.0059±0.0014	28.60±2.61	29.07±3.01	3.74±0.14	4.87±0.73	0.201±0.011	5.60±0.53
PER	74.18±4.05	55.1±2.8	86.0±4.7	0.194±0.036	0.0075±0.0014	26.20±3.01	26.47±3.34	3.92±0.42	4.38±0.58	**0.174±0.012**	**4.60±0.77**
PCL_bv_num	76.24±1.88	58.0±3.7	87.4±2.6	0.175±0.016	0.0058±0.0005	30.21±1.77	30.76±1.61	4.02±0.31	**3.32±0.24**	0.185±0.011	5.26±0.31
PCL_label	72.99±2.33	53.2±4.3	85.0±3.2	0.207±0.021	0.0072±0.0017	29.50±3.63	30.36±4.08	4.05±0.65	4.38±0.67	0.200±0.010	5.65±0.63