Learning-based Bayesian Inference for Testing of Autonomous Systems

Supplementary Material

June 2024

1 Generation of training data without expert knowledge for Sequence EVs

1.1 Importance of expert in generating training data

The lack of expert knowledge hinders the generation of "good quality" environmental variables for falsification. The term "good quality" here, is agnostic to failure and does not depend on whether instances of falsifying examples are present in training data. Rather, it refers to the dataset consisting of "realistic" examples that the user wishes to test the dynamic system on. As we have explained in the subsequent paragraphs, the presence of "good quality" data for training the VAE is a sufficient requirement for our framework.

In the absence of expert information about the quality of data in terms of its realistic quotient, a naive approach can be used to generate the data first, and the user must be consulted to determine whether the quality of scenarios generated by the naive approach appropriately meets the standards of testing needs. In Section-IV B (main manuscript), the approach presented for generating training-dataset is one of the contributions of the paper, to inspire a data generation scheme in the absence of sufficient baseline information, especially for racelines for path-tracking and autonomous racing. Presently, there is no available dataset that we could utilize as a reference for generating realistic racelines that are also appropriate for tracking, given the dimensions of the F1-Tenth vehicle we used. Hence, we developed the technique outlined in Section IV-B to generate the required reference data. This is primarily due to the fact that the proposed learning scheme is supervised, i.e., requires good quality ground-truth demonstrations to learn from. Kindly note that the VAE learning pipeline is primarily to generate diverse racelines with complicated curvatures that imitate the racelines seen in practice.

We have also included a discussion on how the uncertainty of obstacles' configurations can be tackled using over-approximation, which is also relevant to this discussion of efficiency of our approach. Additionally, the discussion pertaining to incorporating agent information in the training process is also relevant here (kindly refer to the section titled "Introducing failure information in training pipeline: Dataset Design Section V-B 2") in our response).

1.2 How to generate data in the absence of approach outlined in Section IV-B

In the absence of our proposed technique or existing datasets for training, one can use a parametric raceline data-generation scheme which generates a set of curves like circles, ellipsoids, etc. This would only limit the diversity of racelines on which we can test our dynamic system and control algorithm, i.e., it affects the quality of generated environmental variables, but does not change the sampling efficiency of various algorithms discussed in the paper, since that depends on sampling volume and dimensionality of the decision variable, which is kept common across all sampling algorithms. Consider for example, a parameterization that does not require learning-based approach, with the environment variable $o \in \mathbb{R}^{N\times 2}, o_k = [x_k, y_k]^\mathsf{T}, o_k = \tilde{o}_k[0:2]$, where $\tilde{o}_k = [x_k, y_k, \psi_k, \omega_k]$ evolves as:

$$\begin{aligned} \omega_k &= a\sin(t) \\ \psi_k &= \psi_{k-1} + \omega_k \Delta t \\ x_k &= x_{k-1} + v_{\text{ref}} \Delta t(\cos(\psi_k)) \\ y_k &= y_{k-1} + v_{\text{ref}} \Delta t(\sin(\psi_k)) \end{aligned} \tag{1}$$

Here $t = k\Delta t$. This representation generates dynamically feasible racelines parameterized by a. Falsifying examples of racelines generated by sampling across $a \in \mathbb{R}$ for $v_{\text{ref}} = 2.7$, using 2^{nd}-order LA are shown in Fig. 1, which involve failure due to divergence or not reaching the goal in desired time. Consider another example of a two-parameter parametric curve for the environment variable $o \in \mathbb{R}^{N\times 2}$ where $o_k = [x_k, y_k]^{\mathsf{T}}$ evolves as:

$$\begin{aligned} x_k &= x_{k-1} + v_{\text{ref}}\Delta t(\sin(a_1 t) + 3\sin(a_2 t)) \\ y_k &= y_{k-1} + v_{\text{ref}}\Delta t(\cos(a_1 t) + 3\cos(a_2 t)) \end{aligned} \tag{2}$$

Here $t = k\Delta t$. Falsifying examples of racelines generated by sampling across values of $a = [a_1, a_2] \in \mathbb{R}^2$ for $v_{\text{ref}} = 2.7$, using 2^{nd}-order LA are shown in Fig. 2, which shows failures due to divergence, which was considered as the only failure mode in this example. Fig. 3 shows falsifying examples generated by sampling across the 4-dimensional latent-search space ($d_2 = 4$) that is learnt using the VAE. The key difference between the naive parameterization (Fig. 1,Fig. 2) and the VAE-based approach (Fig. 3) is in the complexity and diversity of the type of curves generated.

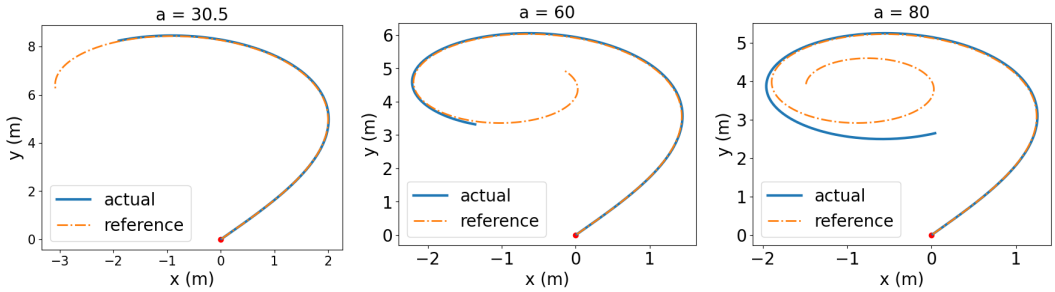

Figure 1: Examples of falsifying racelines obtained using 1 parameter parametrization ($d_2 = 1$)

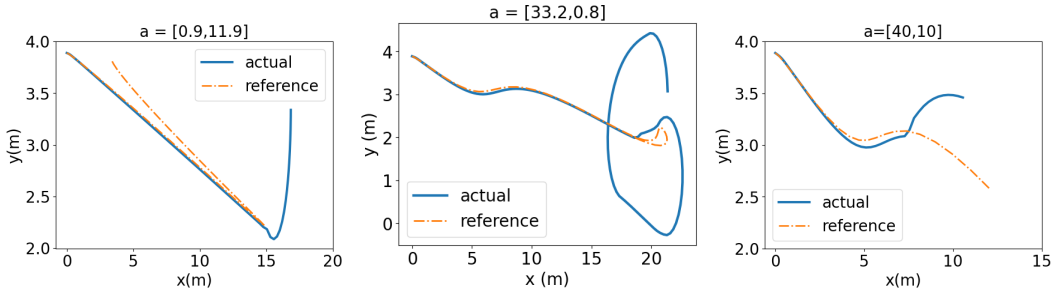

Figure 2: Examples of falsifying racelines obtained using 2 parameter parametrization ($d_2 = 2$)

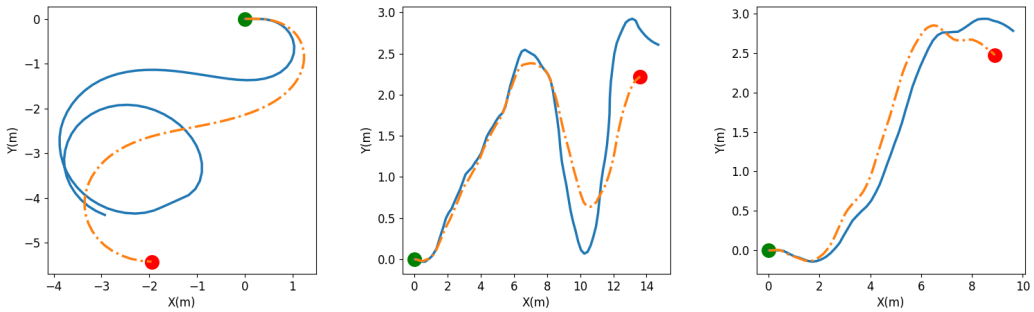

Figure 3: Examples of falsifying racelines obtained using our VAE-based raceline generation method (latent variable parameterization ($d_2 = 4$)

Impact of training data quality on the efficiency of algorithms

We would like to highlight that the contribution of this paper is two-fold: a) learning-based schemes for generating realistic and good quality environmental variables, and b) 2^{nd}-order Langevin sampling for faster exploration of failure modes. In Table-I (main manuscript), Random sampling and Langevin sampling (1^{st} and 2^{nd}-order) are applied to the exploration of latent variables after learning the appropriate latent features using the generated data. The results show that 2^{nd}-order LA is more efficient in sampling in the same lower-dimensional latent space compared to Random Sampling and 1^{st}-order LA.

All the algorithms mentioned in the paper can also be applied directly to explore racelines parameterized by other methods (Fig. 1, Fig. 2), without a compromise on the efficiency of algorithms, as long as the naive parametric racelines are parametrized by lower dimensional variables. Kindly note that for a simpler dataset generation pipeline using the one-dimensional latent parameterization of racelines as shown in Fig. 1, the performance of all sampling algorithms is noted to be comparable to that observed in Case-study 1 (Section V-B, main manuscript), i.e., 2^{nd}-order LA still outperforms Random Sampling in discovering more failures with higher mean and maximum cost of failures, with a comparable dispersion (Table. 1 shows the performance of 1 Random seed with 100 samples for the algorithms discussed in the paper). We have also included the table, along with key points from this discussion for the perusal of our readers on the project website [**?**]. Note that for naive parameterization such as the one shown in Fig. 1 and Fig. 2, the performance difference between different sampling algorithms is marginal, however, 2^{nd}-order LA becomes increasingly efficient as the dimension and complexity of representation increases (Fig. 3).

Table 1: LQR Speed+Steering Control (Bicycle Model) Parametric Raceline generation.

	Failure rate	Mean cost	Max cost	C_{disp}
Random sampling	0.61	33.9	38.07	3.43
1^{st}-order LA	0.17	27	38.70	3.71
2^{nd}-order LA	**0.98**	**36.01**	**38.60**	**3.02**

2 Robustness of our testing method

2.1 Robustness to uncertainty in environment-variable configuration

The robustness of the model's generalization depends on the quality of the training dataset and the loss function used for training. For the prior design of Static EVs, robustness in the model's generalization can be achieved by using a training dataset that overapproximates the actual region of interest. Fig. 4 shows an example of the learnt classification boundary that over-approximates the true boundary. Fig. 5 shows the difference in data collection for generating an over-approximation of the decision boundary. If the obstacle configurations are known with less confidence, for example, the boundaries of the racetrack are not known with full confidence, an over-approximated classification can be learnt. Robustness increases proportionally with over-approximation. The over-approximation can also be used if the distribution of the representation is likely to change and has uncertainty. However, projection on the over-approximated region will be less efficient than the case when the decision boundary can be known with confidence, which is expected. The projected datapoints are shown in Fig. 6 for the nominal case shown in the manuscript and the over-approximated boundary. As long as we know that the racetrack boundary lies in the learnt over-approximated region, the projection method will be robust to uncertainty in obstacle configuration

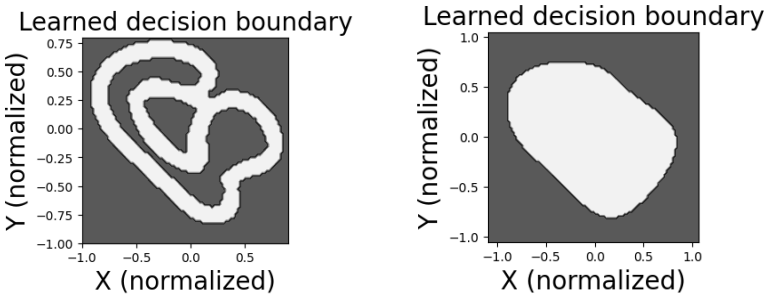

Figure 4: Learnt decision boundaries: Original (left) and Over-approximation (Right)

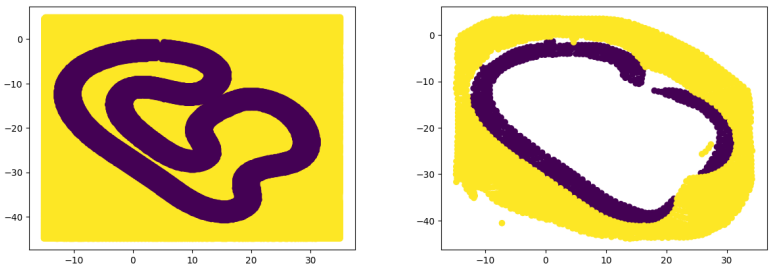

Figure 5: Data collection: Original (left) and Over-approximation (Right). Data in purple (yellow) refers to points on the racetrack (out of racetrack)

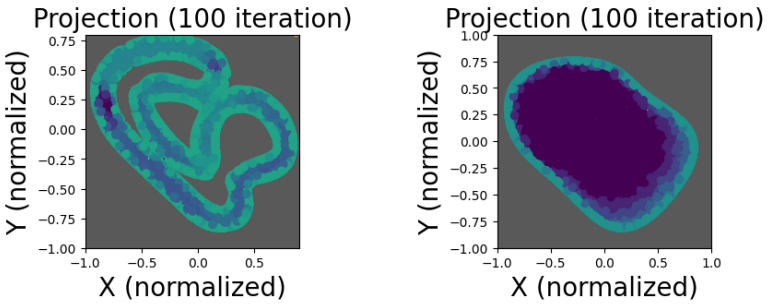

Figure 6: Neural projection implementation: Original (left) and Over-approximation (Right). Note that due to the over-approximated learnt decision boundary, the projection step projects on a much larger region, thereby adding robustness to uncertainty in position of environmental scenarios, but compromising efficiency of the method

For the prior design of Sequence EVs, the VAE learns the representative latent features, and consequently generalizes well if the training dataset is drawn from the same distribution as the true distribution of environmental scenarios. Under this assumption, the best way to ensure robustness is to ensure better diversity of configurations in the training dataset.

2.2 Robustness to uncertainty in agent dynamics

For real-world testing, there are always mismatches between the actual dynamics and the mathematical dynamical models. It is important to ensure robustness of testing methods to uncertainty in agent dynamics. One of the ways this robustness can be incorporated is by including the model mismatches as a bounded noise exogenous to our dynamic system, and performing testing using Bayesian inference as discussed in the paper and observing the robustness of the failure distribution across various levels of noise. Failures that are common across noise are likely to be repeated in real world, whereas those which are sensitive to noise levels would require further analysis. The hardware experiments can be leveraged here to estimate a bounded prior distribution of the noise to generate a failure distribution that closely replicates real world failures. This estimation can be updated iteratively using carefully planned hardware experiments.

3 Details of the cost threshold (c^*) design

The definition of cost function (Section II) does not depend explicitly on the controller, but depends indirectly on it, since X is a function of the chosen controller. Consequently, the threshold c* depends on the controller implemented. The threshold is also task specific, since it depends on the actual range of the cost function, and the cost function itself is modeled in a task-specific manner.

3.1 Guidelines for choosing c^*

In the present work, c^* was chosen by trial and error. In general, c^* can be chosen by fixing a specific value of c^* and observing the range and severity of the failures obtained using it, and accordingly increasing (decreasing) c^* to increase (decrease) the severity of failures. Note here, that the physical interpretation of the severity of failure depends on the definition of the cost function. Additionally, there is inherent subjectivity here because failure is defined as scenarios for which $c > c^*$. Hence, ideally, resulting scenarios corresponding to different thresholds of c^* should be generated. Consequently, based on user feedback an appropriate c^* can be chosen by fine-tuning the threshold. For example, for Case-study 1 (Section V-B, main manuscript), different falsifying trajectories corresponding to various values of c^* are shown in Fig. 7. It can be seen that the divergence of trajectories from reference path increases as c^* increases, which is a metric of severity of failure in this case. Based on these representative falsifying scenarios for various values of c^*, we can choose a c^* and fine-tune it further to generate suitable collection of falsifying examples as per user-needs.

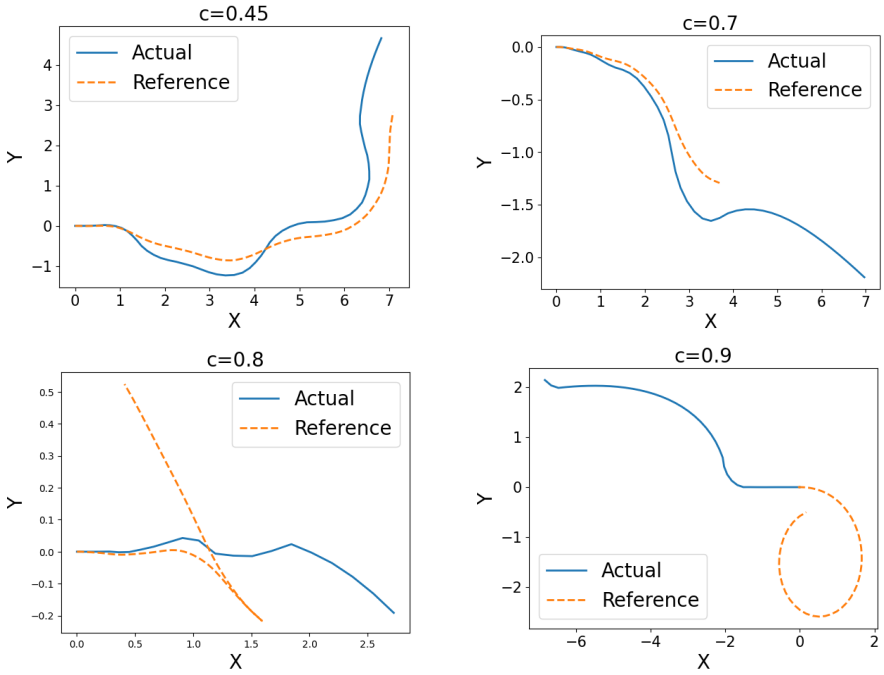

Figure 7: Representative examples of falsifying trajectories with the cost $c = c^*$, where $c^* = [0.45, 0.7, 0.8, 0.9]$. Falsifying examples with higher values of c^* show higher divergence from the reference path.

4 Ways to enhance the network's reliability in learning environment variables correspond to failure

In our presented approach, we rely on the training pipeline for learning unbiased prior distributions of the environment variables, and the discovery of failures is executed in the cost-guided exploration, where we make use of the agent's dynamics, actuators and controller as a part of the differentiable simulator to infer the posterior. As an example, in the Static-EV case, we use the Bayesian inference framework with a NN classifier trained on unbiased data, and achieve efficient performance (Table-II, Table-III in the main manuscript), where the learnt prior does not depend on any failure information of the system. There are several ways of incorporating the agent's information in the training pipeline, which we have discussed in detail below.

Introducing failure information in training pipeline: Dataset Design Section V-B 2)

Incorporation of failure information in training dataset can lead to learning representations that are biased towards specific failure modes. In the specific example of Case-study 1, in addition to implementing the pipeline presented in Section-IV B, we chose to intentionally generate raceline dataset that would result in failure. As noted in Section V-B 2), this was done to show that a skewed dataset that results in a specific kind of failure-mode will be dominant in the learnt representation of the VAE (Fig 5 in the paper), which we discuss in the Introduction (Section I), as one of the shortcomings of directly using learning-based representations in testing without cost-guided exploration. As we have explained in the previous response, this was done to assess and demonstrate the efficient exploration capabilities of 2^{nd}-order LA. While 2^{nd}-order Langevin is able to discover failure modes far away from this region even in the presence of a biased prior, this problem can get more complicated when the learning representation carries information about multiple failure modes, generating a multi-modal prior distribution. This can also affect the diversity of the failures inferred in the posterior.

By introducing failure information in the training dataset, such as input constraints, we can enhance the efficiency of the learnt prior, however, this is recommended only if it is known a-priori that other failure modes do not exist.

Introducing failure information in training cost

The training cost can be manipulated to encourage failure due to input constraints, or other physical phenomenon related to the agent's dynamics or controller. This may have a similar effect as introducing failure information in training dataset, and may lead to over-weighting of some failure causes over the others in the learnt representation. This would require hyperparameter tuning to ensure that all failure modes are appropriately prioritized in the learning pipeline.

Our framework remains agnostic to the "cause" of failures while searching for falsifying examples. This is done to ensure that with enough samples, failures corresponding to all possible "causes" will be revealed while analysing the failure data gathered from sampling, regardless of the number of failures corresponding to a particular cause. To embed the causes of failure pertaining to agent's dynamics, actuators and the controller would require a different framework design from the one we have considered in this paper, to ensure diversity of failure modes.