# Runtime Monitors for Operational Design Domains of Black-Box ML Models

**Hazem Torfah**
University of California at Berkeley
torfah@berkeley.edu

**Sanjit A. Seshia**
University of California at Berkeley
sseshia@berkeley.edu

## Abstract

Autonomous systems are increasingly relying on machine learning (ML) components to perform a variety of complex tasks in perception, prediction, and control. To guarantee the safety of ML-based autonomous systems, it is important to capture their operational design domain (ODD), i.e., the conditions under which using the ML components does not endanger the safety of the system. In this paper, we present a framework for learning runtime monitors for ODDs of autonomous systems with black-box ML components. A runtime monitor of an ODD predicts based on a sequence of monitorable observations whether the system is about to exit its ODD. We particularly investigate the learning of optimal monitors based on counterexample-guided refinement and conformance testing. We evaluate our approach on a case study from the domain of autonomous driving.

## 1 Introduction

In recent years, there has been an increase in using autonomous systems in various safety-critical applications. Operating in complex environments, modern autonomous systems depend on machine learning (ML) techniques to solve complex tasks in perception, prediction, and control [2, 4, 17]. ML components such as deep neural networks are, however, brittle; unanticipated changes in the environment may cause a neural network to produce faulty outcomes endangering the safety of the autonomous system [1, 7, 9]. A major driver for this brittleness, is the distribution gap between the data used in training and the data received at runtime. Out-of-distribution learning is key to reducing this gap and has been subject to extensive investigation in the ML community [10, 14, 15, 23, 24]. Out-of-distribution learning aims to detect shifts in the distributions of inputs and labels of ML models, with the goal of learning models that generalize well on unseen inputs. It thus treats the problem on the component level. However, in autonomous systems, ML components are part of a larger system and their behavior is influenced by many external factors, some of which may not have been (sufficiently) covered during training. Analyzing the module in isolation does not tell us anything about its behavior under external factors. It is, therefore, crucial to also examine ML components in the larger context of their systems and learn how they affect their safety. Specifically, it is important to capture the system-level conditions under which these components are guaranteed to maintain the safety of the overall system, also known by its *operational design domain* (ODD).

In this paper, we introduce a simulation-based approach for learning *monitorable operational design domains* of black-box systems, in particular those with critical ML components. We emphasize the role of monitorability in this context. To assure the safety of the system, the boundaries defined by an ODD must be monitored during operation and the system should only operate (autonomously) using the ML component when these boundaries are met. In contrast to general notions of ODDs [3, 12, 13, 16, 19], we are, therefore, interested in learning ODDs defined over a monitorable feature space and which can be implemented as runtime monitors. Not every ODD can be monitored at

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

runtime. Some aspects of the ODD may not be reliably observable or are too expensive to observe. A monitor relying on these aspects is not (efficiently) implementable.

In many cases, the system or its ML components are black boxes with no access to their implementation or the data used in training. To learn a monitor that accurately represents the ODD, data used for learning the monitor needs to provide good coverage of the feature space over which the ODD is defined. We emphasize that a powerful ODD needs to be specified over runs of the system, as the history of observable features allows us to approximate hidden states of the black box system. To scale to this complex high-dimensional feature space, our framework builds on VERIFAI, an open-source toolkit for the formal design and analysis of systems that include ML components [6]. VERIFAI operates on an abstract semantic feature space, representing different spatial and temporal configurations of objects and agents, in which we want to analyze a system. This space is represented using SCENIC, a powerful probabilistic programming language for modeling environments [8]. A SCENIC program defines a distribution over configurations of physical objects and their behaviors over time from which various simulations can be sampled. While running the system in a sampled configuration, runs over valuations of the ODD's feature space are collected and evaluated according to a (temporal) system-level specification. These form the training set for learning the monitor.

In the following sections, we briefly introduce the learning framework and show how it can be used to learn monitors for the ODD of an autonomous car with a black-box perception module for lane keeping. We particularly show the importance of learning *state-based* monitors that base their predictions on a history of observations [20] and discuss the challenges in learning such monitors. For more information on the formal definition of monitorable ODDs, the learning problem, and technical details on our framework, we refer the reader to [22].

## 2 Framework

Our framework is shown in Figure 1 and is composed of three main components: *simulation-based analysis*, *data generation and learning*, and *conformance testing*. For simulation-based analysis, we use VERIFAI [6]. Given an executable model of the system with the black-box (ML) component, a model of the environment in which the system is to be executed, given as a probabilistic program written in SCENIC [8], we use VERIFAI to run simulations and evaluate them according to a provided system-level specification i.e., one defining a property of the system (For example, an autonomous car should never exit its lane). The evaluated simulations are then forwarded to another component for data generation. The data generation component performs several operations on top of the simulation traces, applying certain filters, transformations, and slicing (see, e.g., [21, 22]). Once the data has been prepared for learning, a learner of our choice runs on top of the data. The outcome is an (optimal) monitor implementing the ODD. A monitorable ODD in our framework is learned in terms of a desired class of programs defined over the observable feature space. For the given system-level specification, one that defines a general ODD over a possible non-observable abstract feature space, our framework can be used to learn a monitorable ODD from the class of programs that predicts whether the system will violate the specification, based on sequences of valuations of the observable feature space. In our case study, we fix the class of decision trees as the class of monitors and learn two types of decision tree monitors, stateless monitors defined over positional values of the feature space (sequences of length 1), and state-based monitors defined over sequences of feature valuations of length 20. Finally, a conformance tester checks the quality of the monitor. Here, the conformance tester may use further simulation runs, using VERIFAI, to search for any counterexamples. If conformance testing succeeds, the framework terminates and returns the so-far learned monitor. Otherwise, counterexamples found during testing are forwarded to the data generating process to compute a new set of data over which a new monitor is learned. Counterexamples can be particularly of two types, *false positives* and *false negative*. Learning can be biased toward one of these types by changing the value $w_{fn}$.

## 3 Case Study: Image-based Lane Keeping

We used our framework in an experiment for learning a monitor for the ODD of an autonomous vehicle with an image-based perception module used for lane keeping. The perception module uses a convolutional neural network (CNN) that, for a given snapshot taken by a camera mounted at the front of the vehicle, returns the estimated cross-track error to the centerline of the lane. We are interested in learning a monitor which based on the values of the features of precipitation, cloudiness, the sun

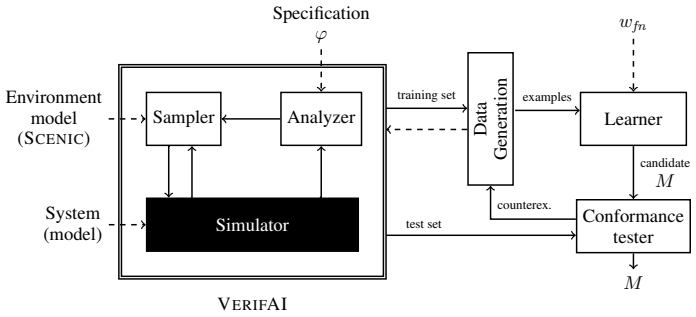



Figure 1: Extension of VERIFAI with the ODD learning framework

```
param weather =
        Uniform('ClearNoon', 'CloudyNoon',
                'WetNoon', 'MidRainyNoon',
                'ClearSunSet')

lane = Uniform(*network.lanes)
start = OrientedPoint on lane.centerline

ego = Car at start,
        with visibleDistance 60,
        with behavior EgoBehavior(10)
```

Figure 2: Scenic program and corresponding sampled scenes

angle, location on the map (junction or straight road), and radar data covering the front spectrum of the vehicle, determines whether the vehicle will remain in the lane. In the following, we provide some details on the experimental setup and results.

### 3.1 Setup

Our setup uses VERIFAI's interface to the CARLA simulator [5]. The perception module is executed as part of a closed-loop system whose computations were sent to a client running inside CARLA. These are named values that represent the simulator state, such as the position of the car, its velocity, heading, weather conditions, other objects on the road.

The environment is modelled by the SCENIC program depicted in Figure 2. The sampler chooses simulations in different weather conditions, roads and initial positions on the road, and sun angles, thus, sampling different times of the day and their corresponding shadowing effects. The behavior of the ego car is implemented as a call to an external function implementing `EgoBehavior`, which defines a Simplex-based architecture [18] that depending on the decision of a learned monitor switches between the ML perception-based control and a safe controller.

The simulation runs were evaluated based on the temporal specification of "never invading another lane", using a built-in CARLA specification for detecting lane invasions. Initially, we obtained the training data from 115 simulations[1]. In each conformance testing round, we used another 115 i.i.d simulations sampled from the SCENIC program. Lastly, we fixed decision trees as the class of our monitors and used a decision-tree learning procedure provided by the sci-kit learning library[2].

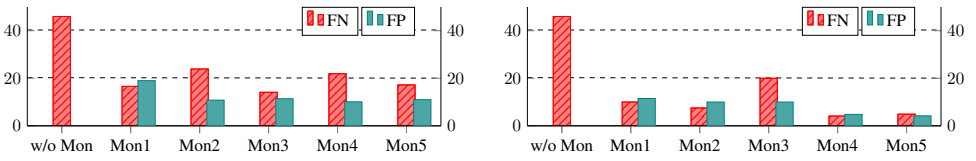
All experiments were conducted on a machine with a 3.5GHz 10-Core CPU, 64GB of RAM, and a GPU with 6 GPCs and a total 3072 cores.

### 3.2 Experiments

We conducted two types of experiments. In the first experiment, we learned stateless monitors that predict a violation of the system-level specification solely on the current valuation of the ODD's

[1]The number of simulations was computed using Hoeffding's inequality [11] for confidence value $\alpha = 0.01$ and error-margin $\epsilon = 0.1$.

[2]`https://scikit-learn.org/`




(a) Stateless monitors learned over only current evaluations of features.

(b) State-based monitors learned sequence evaluations of features of length 20 and a prediction horizon of offset 20.

Figure 3: Statistics of stateless and state-based monitors learned over five iterations. The false negatives are represented in lined red and the false positives in solid green.

feature values. In the second experiment, we learned state-based monitors over sequences of feature valuations of length 20 steps. The labels of the training set were computed based on the recorded value of the system-level specification 20 steps into the future.

Figure 3 shows a statistical evaluation of all the monitors learned in the two experiments over five learning iterations. In each iteration, we computed the rate of false negatives (shown in lined red) and false positives (shown in solid green). The false negatives and false positives were computed by running the system in a non-Simplex fashion, i.e., without interference by the monitor, yet recording the predicted values of the monitor. If the monitor predicted a violation, yet no violation happened in 20 steps in the future, then this was marked as a false positive. If the monitor did not predict a violation, yet one occurred in 20 steps, then this was marked as a false negative. The result for the "w/o monitor" case, was computed by the rate of specification violations over all simulations.

All monitors learned using our framework were able to significantly increase the safety of the autonomous vehicle (decreasing the rate of false negatives) by switching to a safe controller when the CNN is not trusted to keep the system safe. A significant decrease in the false negatives rate is noticeable for state-based monitors in comparison to stateless monitors (compare false negative rates in Figure 3(a) and Figure 3(b)). With larger histories, the complexity of the monitors however increases. When learning a monitor, it is therefore important to balance the length of input sequences to the monitor and the delay caused by evaluating the monitor on this sequence length. For example, the positional monitors learned in Figure 3(a) caused an average delay of 201 $\mu$s in each computation step. Using state-based monitors resulted in an increase in the average delay to 315 $\mu$s.

In general, after a certain number of iterations, we will exhaust the feature space of the ODD. When the rates of false positives and false negatives fluctuate within a certain bounded range, then this is an indication, that we reached optimal monitors (in a statistical sense) with respect to the ODD's feature space. This was already clear after the third iteration when learning stateless monitors. For state-based monitors, the boundaries started to stabilize after iteration 4.

Lastly, integrating the learned monitors into the system resulted in the stabilization of the operation of the vehicle along the road (the results are based on a safe controller that is close to perfect and always guides the vehicle to the center of the lane). Monitors with smaller false negatives rates have a lower cross-track error (CTE) range than those with larger ones, especially, the system that uses no monitor. A lane invasion was recorded every time the vehicle exited the range [-.6, 0.6]. While the system solely using the CNN had an average CTE range of [-1.17,1.16], with state-based monitors we were able to reduce this to the range of [-0.89,0.68] (based on the statistics of Mon4). The stateless monitors on the other hand did not achieve this range.

As with any learning problem, a good outcome requires choosing the right learning parameters. In the case of learning monitors for ODDs, this highly depends on finding the right length of sequences, the prediction horizon, and most importantly the features used in learning the ODD. In the future, we plan on exploring these aspects more, building on experiences in counterexample-guided learning methods to obtain good counterexamples in each iteration of the learning process.

## 4 Conclusion

We presented a framework for learning monitorable operational design domains for autonomous systems that include ML components. The framework is based on a counterexample-guided learning approach that builds on simulation-based analysis methods to retrieve its counterexamples. We evaluated our approach on a lane keeping case study from the domain of autonomous driving and showed the importance of state-based monitors in comparison to monitors that compute their prediction based on positional data.

## Acknowledgments and Disclosure of Funding

The authors are grateful to Daniel Fremont for his contributions to the VERIFAI and SCENIC projects, and assistance with these tools for this paper. The authors want to thank Sebastian Junges and Marcell Vazquez-Chanlatte for the insightful discussions and their feedback on this project. The authors also want to thank Johnathan Chiu, Tommaso Dreossi, Shromona Ghosh, Francis Indaheng, Edward Kim, Hadi Ravanbakhsh, Ameesh Shah, Kesav Viswanadha and Carol Xie for their valuable contributions to the VERIFAI project.

This work is partially supported by NSF grants 1545126 (VeHICaL), 1646208 and 1837132, by the DARPA contracts FA8750-18-C-0101 (AA) and FA8750-20-C-0156 (SDCPS), by Berkeley Deep Drive, by C3DTI, by the Toyota Research Institute, and by Toyota under the iCyPhy center.

## References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul W. Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Inf. Fusion*, 76:243–297, 2021.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[3] Marjory S. Blumenthal, Laura Fraade-Blanar, Ryan Best, and J. Luke Irwin. *Safe Enough: Approaches to Assessing Acceptable Safety for Automated Vehicles*. RAND Corporation, Santa Monica, CA, 2020.

[4] Thomas G. Dietterich and Eric Horvitz. Rise of concerns about AI: reflections and directions. *Commun. ACM*, 58(10):38–40, 2015.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[6] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *CAV (1)*, volume 11561 of *LNCS*, pages 432–442. Springer, 2019.

[7] Tommaso Dreossi, Somesh Jha, and Sanjit A. Seshia. Semantic adversarial deep learning. In *CAV*, volume 10981 of *LNCS*, pages 3–26. Springer, 2018.

[8] Daniel J. Fremont, Edward Kim, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification and data generation, 2020.

[9] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks. *CoRR*, abs/2107.03342, 2021.

[10] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[11] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[12] Patrick Irvine, Xizhe Zhang, Siddartha Khastgir, Edward Schwalb, and Paul Jennings. A two-level abstraction ODDdefinition language: Part i*. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, page 2614–2621. IEEE Press, 2021.

[13] Siddartha Khastgir, Stewart A. Birrell, Gunwant Dhadyalla, and Paul A. Jennings. Calibrating trust through knowledge: Introducing the concept of informed safety for automation in vehicles. *Transportation Research Part C: Emerging Technologies*, 2018.

[14] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *CoRR*, abs/1807.03888, 2018.

[15] Lawrence Neal, Matthew L. Olson, Xiaoli Z. Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VI*, volume 11210 of *Lecture Notes in Computer Science*, pages 620–635. Springer, 2018.

[16] SAE on Road Automated Driving Committee et al. SAE J3016. taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. Technical report, Technical Report.

[17] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Toward verified artificial intelligence. *Communications of the ACM*, 65(7):46–55, 2022.

[18] Lui Sha. Using simplicity to control complexity. *IEEE Softw.*, 18(4):20–28, 2001.

[19] Eric Thorn, Shawn C. Kimmel, and Michelle Chaka. A framework for automated driving system testable cases and scenarios. 2018.

[20] Hazem Torfah. Stream-based monitors for real-time properties. In *RV*, volume 11757 of *LNCS*, pages 91–110. Springer, 2019.

[21] Hazem Torfah, Sebastian Junges, Daniel J. Fremont, and Sanjit A. Seshia. Formal analysis of AI-based autonomy: From modeling to runtime assurance. In Lu Feng and Dana Fisman, editors, *Runtime Verification - 21st International Conference, RV 2021, Virtual Event, October 11-14, 2021, Proceedings*, volume 12974 of *LNCS*, pages 311–330. Springer, 2021.

[22] Hazem Torfah, Carol Xie, Sebastian Junges, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. Learning monitorable operational design domains for assured autonomy. In Ahmed Bouajjani, Lukás Holík, and Zhilin Wu, editors, *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings*, volume 13505 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2022.

[23] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *CoRR*, abs/2110.11334, 2021.

[24] Ying Zhang, Baohang Zhou, Xiaoke Ding, Jiawei Ouyang, Xiangrui Cai, Jinyang Gao, and Xiaojie Yuan. Adversarially learned one-class novelty detection with confidence estimation. *Inf. Sci.*, 552:48–64, 2021.