Out-of-Distribution Dynamics Detection: RL-Relevant Benchmarks and Results

Mohamad H. Danesh [1] **Alan Fern** [1]

Abstract

We study the problem of out-of-distribution dynamics (OODD) detection, which involves detecting when the dynamics of a temporal process change compared to the training-distribution dynamics. This is relevant to applications in control, reinforcement learning (RL), and multi-variate time-series, where changes to test time dynamics can impact the performance of learning controllers/predictors in unknown ways. This problem is particularly important in the context of deep RL, where learned controllers often overfit to the training environment. Currently, however, there is a lack of established OODD benchmarks for the types of environments commonly used in RL research. Our first contribution is to design a set of OODD benchmarks derived from common RL environments with varying types and intensities of OODD. Our second contribution is to design a strong OODD baseline approach based on recurrent implicit quantile networks (RIQNs), which monitors autoregressive prediction errors for OODD detection. Our final contribution is to evaluate the RIQN approach on the benchmarks to provide baseline results for future comparison.

1. Introduction

In many machine learning applications, the environment may change in various ways between training and testing. For example, in RL, real-world training can be costly, necessitating simulation and transferring to the real world. Even when RL training and/or validation is possible in the real environment, properties of the environment may change after deployment, e.g., due to mechanical wear, weather conditions, or the behavior of other actors. When such differences between training and deployment arise, a trained policy can behave unpredictably, which is undesirable or even dangerous in many applications. One approach addressing environment change is to follow a robust control framework (e.g., (Abdallah et al., 1991; Sage et al., 1999)), where the aim is to be robust to certain classes of variation. However, in reality, environments can change in unknown ways that cannot be anticipated at design time. Thus, it is important to be able to detect such changes during operation so that appropriate measures be taken to ensure safety and avoid large degradation in performance.

In this paper, we focus on detecting changes in environment dynamics, a problem we refer to as out-of-distribution dynamics (OODD) detection. Specifically, motivated by the challenges of deploying RL controllers, we consider OODD for trained RL controllers operating in common RL-benchmark environments. Our first contribution is to create a set of OODD benchmarks, which contain trajectories under nominal and anomalous dynamics, where different dynamics-changing anomalies are introduced at random times. The objective is to detect when an anomaly with respect to the nominal trajectories is introduced in the trajectories while maintaining a low false-positive rate.

Our second contribution is to design an OODD detection approach, based on recurrent implicit quantile networks (RIQNs), to serve as a strong baseline. This model forms an anomaly signal by comparing auto-regressive predictions of future observations to the actual observations. Varying the horizon used for the auto-regressive predictions can impact the types of dynamics changes that can be effectively detected as well as the delay in detection. Our third contribution is to evaluate the RIQN approach on the benchmarks using different horizons to provide baseline results on the overall AUC of OODD, as well as the detection delay and corresponding false alarm rate (FAR). We hope that this work will inspire new OODD approaches and extensions to the benchmarks and evaluation metrics/protocols in RL[1].

[1]School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA. Correspondence to: Mohamad H. Danesh <daneshm@oregonstate.edu>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

[1]The benchmark code:
https://github.com/modanesh/anomalous_rl_envs
The baseline code:
https://github.com/modanesh/recurrent_implicit_quantile_networks

2. RL-Relevant OODD Benchmarks

The OODD benchmarks are derived from RL benchmark environments; classic control from OpenAI Gym (Brockman et al., 2016) (Acrobot, Cartpole, and LunarLander), physics simulations from Bullet (Coumans & Bai, 2016–2021) (Ant, Hopper, HalfCheetah, and Walker2D), and an F16 flight simulation including autopilots for fixed airspeed and fixed altitude, and a ground collision avoidance system (GCAS). More details on the F16 environment are provided in the Appendix B. In all environments, the observations used for OODD are continuous environment state features, which vary in dimension depending on the environment, e.g., CartPole, Ant, and F16 have 4, 28, and 17 features, respectively. To obtain controllers for the environments, we trained policies via DQN (Mnih et al., 2013) and TD3 (Fujimoto et al., 2018) RL algorithms for OpenAI Gym and Bullet, respectively. For the F16 setting, we used the provided autopilots and GCAS. Using these controllers, we generated a set of 10,000 nominal trajectories using the default environment settings. These trajectories are intended to represent the training environment to be used to train OODD detectors.

For each environment, we generate various types of anomalous trajectories which follow the nominal dynamics until a random time point upon which there is a switch to anomalous dynamics. The anomalous dynamics correspond to different ways of modifying the state features provided to the controllers through sensors. Note that such modifications can impact the behavior of the controllers, which in turn can impact the evolution of the trajectories. Importantly, we avoid modifications that cause a controller to completely fail (e.g., falling or crashing) soon after the anomaly is introduced. This is because such failures are easy to detect and do not allow testing the ability to detect anomalies before "disasters", which is a primary goal. To achieve this, we only modify a small percentage of the features (at most 20%) chosen for each environment based on tests with the trained controllers. Each anomalous trajectory is generated by randomly selecting the appropriate size subset of features to modify, which provides diversity across the anomalous trajectories. We consider the following four types of anomalous trajectories, which differ in the way they modify the selected features.

IID Noise. Gaussian noise is added to the features. For each environment, the mean and standard deviation is selected to avoid near-term failure of the controller.

Sensor Shutdown. The output of the sensor changes to zero immediately after the anomaly and stays at zero for the rest of the run. If applied to the most important features for a policy, it could easily degrade the performance. It resembles the case where a sensor fails and needs to be replaced.

Sensor Calibration Failure. It multiplies the sensor's output by a constant throughout the anomalous run. Similar to the IID noise, for each case study, there is a specific pre-determined constant.

Table 1. Details of different anomalous environment modifications.

Environment	IID		Calibration failure	Sensor drift
	Mean	STD		
Acrobot	2	2	10	1/200
CartPole	1	2	3	1/5000
LunarLander	0	1	3	1/10000
Ant	0	1	2	1/10000
HalfCheetah	0	0.12	1.25	1/50000
Hopper	0	1	1.25	1/50000
Walker2D	0	0.12	1.25	1/50000
Fixed Airspeed	1	0	3	1/100
Fixed Altitude	1	0	3	1/100
GCAS	1	0	3	1/100

Sensor Drift. Relative to the time step in a trajectory, a small amount of noise is injected into the chosen sensor. As the trajectory progresses, the magnitude of the injected noise increases.

Table 1 provides relative details for each environment and anomaly type. Table 2 gives the average reward of 1,000 trajectories achieved by controllers in the nominal and anomalous environments. We see a decrease in reward for anomalous environments but not catastrophic levels of decrease.

3. Prediction-Based OODD Baseline

In this section, we develop a straightforward OODD detection approach based on *anomaly detection via prediction errors*, which involves the following steps: 1) train a dynamics model that can predict future observations based on past observations, 2) at each time point t use the model to make auto-regressive predictions of observations at time $t + \Delta$, possibly for multiple values of Δ, 3) generate an anomaly signal based on the difference between the predictions and actual observations. Note that when used for real-time anomaly detection, the value of Δ places a fundamental lower-bound on how much delay there must be before the onset of an anomaly is detected (i.e., a delay of at least Δ). However, for "slower moving" anomalies (e.g., subtle sensor drift), it may be necessary to use larger values of Δ to detect an anomaly's impact. Below we first describe our predictive model, followed by how it generates anomaly signals for OODD.

Recurrent Implicit Quantile Networks. To capture aleatoric uncertainty, we ideally want a model predicting a distribution over future observations. For this purpose, we draw on Implicit Quantile Networks (IQNs) (Dabney et al., 2018), which were proposed to represent value-distributions

in distributional RL. IQNs learn an implicit representation of a value or feature distribution using the quantile Huber loss. At a functional level, once trained, it can be used to generate samples from the learned distribution conditioned on the input. More discussion of the architecture is in Appendix D, and we refer the reader to the original IQN paper for further details. For our work, we modify the original IQN approach in three ways: 1) since we are interested in dynamics prediction in problems where observations do not fully capture the state, we add recurrent memory to the IQN network via a GRU (Cho et al., 2014) layer resulting in the recurrent IQN (RIQN) model[2], 2) rather than predicting action values from state-action pairs as input, we train an RIQN for each observation feature to predict the feature distribution at time t based on the previous values of the features, and 3) rather than training via noisy RL-derived target values, we follow a supervised learning approach.

More specifically, for supervised learning, we minimize an asymmetric variant of the Huber loss (from prior work (Dabney et al., 2018)) on predictions of each feature value at time t given all observation before time t. In addition, since we use the RIQN to generate auto-regressive predictions of future observation sequences, we potentially face the problem of compounding prediction error. That is, since during auto-regressive prediction, the network's outputs are fed back to the input, error accumulation can occur at test time. To address this, we use the Scheduled Sampling approach (Bengio et al., 2015), which conducts training using inputs from ground truth and auto-regressive samples.

Anomaly Detection with RIQNs. Given an RIQN model and a sequence of observations $f_{1:t}$ up to time t, we can auto-regressively sample a future predicted sequence of length H, $\hat{f}_{t+1:t+H}$. First, sample $\hat{f}_{t+1}$ given $f_{1:t}$, followed by sampling $\hat{f}_{t+2}$ given $f_{1:t}, \hat{f}_{t+1}$, and so on until sampling a value for $\hat{f}_{t+H}$. By repeating this process we can generate multiple sampled sequences that for any time $t+\Delta$ yields a set of samples $\{\hat{f}^{(i)}_{t+\Delta}\}$ that can be used to represent the predictive distribution at time $t+\Delta$.

The above auto-regressive sampling approach provides a way of capturing aleatoric uncertainty in a dynamic process using a single RIQN model. To account for epistemic uncertainty due to limited nominal data, we train an ensemble of RIQN models, each of which can be used to sample predicted observation sequences. Each model is trained starting from different initial conditions and different hyperparameter settings, attempting to encourage diversity. In this work, we did not attempt to optimize the ensemble training approach, nor did we attempt to quantify how well the ensembles capture epistemic uncertainty.

[2]In Appendix C we compare results using the RIQN model versus a non-recurrent IQN (N-RIQN) model, which demonstrate significant benefits to including memory.

Given an ensemble of e RIQN models, a observation sequence $f_{1:H}$, and a detection horizon Δ, we can now define a simple anomaly score to assign to any time $t \in \{\Delta, \ldots, H\}$, denoted by A_t. First, use each RIQN model to produce a set of M auto-regressive samples at time t given the subsequence $f_{1:t-\Delta}$, resulting in a set $\{\hat{f}^{(1)}_t, \ldots, \hat{f}^{(M\times e)}_t\}$ of $M \times e$ samples. Given these samples the anomaly score is given by: $A_t = \frac{1}{M\times e}\sum_i \left| f_t - \hat{f}_t \right|_1$, which is just the average L1 distance between the samples and the actual observation at time t. There are other ways to define an anomaly score given an RIQN ensemble, and we leave a deeper comparison for future work. Given an anomaly score for each time point, one can consider various mechanisms for selecting an alarm threshold over which an OODD alarm is raised. In our experiments, we consider different choices for Δ and compare the OODD performance.

4. Experimental Results

Our experiments aim to address the following questions: *1) What is the baseline performance of our RIQN ensemble on the OODD benchmarks? 2) How do the detection horizon Δ and sampling size of the distribution affect the performance? 3) What is the delay in detecting anomalies? 4) How much does recurrent memory impact the performance? 5) How do ensembles impact performance?* We cover a subset of these in the main text. Analysis of sampling size, delay, memory, and ensembles are provided in Appendix C.

Unless otherwise specified, our experiments use an ensemble of 5 RIQN models and a sampling size of 8 per ensemble member. We average all of our performance metrics over 1,000 test trajectories for each anomaly type investigated.

AUC Performance. Once anomaly scores have been calculated for each time point of a test sequence according to Section 3, we can compute the AUC of those scores with respect to the ground truth knowledge of when an anomaly occurred. Table 2 shows results for detection horizons $\Delta = 1$ and $\Delta = 10$ for the different environments averaged over the 4 different anomaly types. We see that, with respect to AUC averaged over anomaly types, the performance for $\Delta = 1$ and $\Delta = 10$ are quite similar and that the baseline RIQN approach is quite strong for these benchmarks. However, there is still apparent room for non-trivial performance improvements via other techniques.

Detection Delay. In many applications, it is important to not just provide anomaly scores, but to also raise an alarm when an anomalous shift in dynamics is suspected. The alarms should have minimal delay to allow for time-critical interventions when necessary. A small delay, however, must be traded-off with the false alarm rate (FAR), which must be sufficiently low for practical applications. Here, we use the sequence of anomaly scores for an observation sequence to

Table 2. Anomaly detection results with $\Delta = 1, 10$ for each environment averaged over the 4 different types of anomalies.

Environment	Policy	Policy Performance		$\Delta = 1$			$\Delta = 10$		
		Nominal Env	Anomalous Env	AUC	Delay	FAR	AUC	Delay	FAR
Acrobot	DQN	-78.78	-95.8	95.1	2.5	0.024	93.2	11.4	0.029
CartPole	DQN	500	430.8	91	2.3	0.035	93.9	11	0.04
LunarLander	DQN	213.55	139.2	91	3.3	0.048	91.8	12	0.05
Ant	TD3	3297.21	2163.6	94.4	2.1	0.03	96	11	0.037
HalfCheetah	TD3	2820.91	2062	92	2.3	0.05	91.6	11.1	0.048
Hopper	TD3	2678.05	1876.6	91.6	2	0.04	91	11	0.041
Walker2D	TD3	2240.81	1634.2	92.2	2	0.044	90.5	11.1	0.04
Fixed Airspeed	-	201.25	192.04	92.3	4.2	0.03	93.3	15.1	0.038
Fixed Altitude	-	265.99	185.28	91.1	3.7	0.07	93.2	14	0.068
GCAS	-	225.16	201.42	91	4.5	0.04	90.4	13.3	0.043

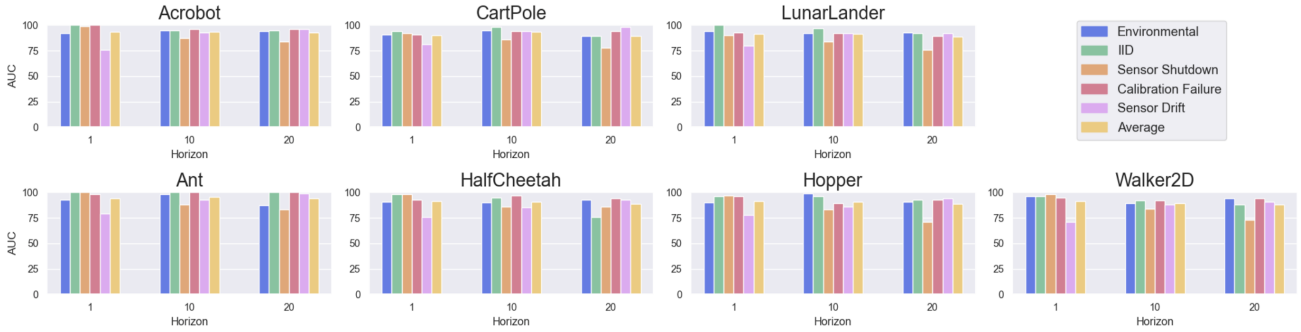

Figure 1. How changing the horizon affects anomaly detection in terms of AUC.

produce alarms using the classic cumulative sum (CUSUM) method (Page, 1954). We used a CUSUM threshold of 0.01 and drift of 0.0018. The threshold sets the amplitude value for the change in the data, and the drift term prevents any change detection in the absence of change. Quantitative results for the average delay and FAR across anomaly types for each environment are reported in Table 2. Note that larger Δ values tend to have larger delays unless they provide significantly better anomaly scores. This is the case when considering performance averaged across anomaly types where $\Delta = 1$ has a much lower delay than $\Delta = 10$. However, we will see below that for certain types of anomalies larger Δ values do pay off.

The FAR values are also similar for the different detection horizons and range from 3% to a maximum of 6.8%. These FAR rates are likely too large for many practical applications as they would result in too many false alarms. Thus, there appears to be significant potential to improve these baseline FAR results while maintaining similar or better delays.

Impact of Horizon. Depending on the anomaly, one could choose to perform OODD detection on an arbitrary horizon. Here, we consider the performance of our method for the individual types of anomalies when the horizon is changed. To do so, we have provided AUC results for $\Delta = 1, 10, 20$ in Figure 1 for the different anomaly types. For the sensor-drift anomalies, which are slowly evolving, we might expect longer horizons to perform better up to a point. The results agree with this expectation and show an advantage for the larger horizons. In contrast, for sensor shutdown, which has a sudden impact, it is expected that shorter horizons will perform better. Figure 1 shows that $\Delta = 1$ has a much better AUC compared to $\Delta = 10, 20$. For other types of anomalies, results among different horizons are quite similar. These results indicate the importance of varying the qualitative types of anomalies in OODD benchmarks to highlight differences in approaches and parameters.

5. Summary

We introduce a set of RL-relevant benchmarks for OODD, with the goal of setting the stage for follow-up research on OODD approaches, evaluation, and benchmark extensions. In addition, we proposed a straightforward baseline approach, based on an ensemble of RIQN models, which provides non-trivial performance on the benchmarks, but leaves apparent room for improvement.

References

Abdallah, C., Dawson, D. M., Dorato, P., and Jamshidi, M. Survey of robust control for rigid robots. *IEEE Control Systems Magazine*, 11(2):24–30, 1991.

Aggarwal, C. C. High-dimensional outlier detection: the subspace method. In *Outlier Analysis*, pp. 149–184. Springer, 2017.

An, J. and Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*, 2015.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chalapathy, R., Menon, A. K., and Chawla, S. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

Chauhan, S. and Vig, L. Anomaly detection in ecg time signals via deep long short-term memory networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–7. IEEE, 2015.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*, 2018.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., and Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Ham, J., Lee, D. D., Mika, S., and Schölkopf, B. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 47, 2004.

Heidlauf, P., Collins, A., Bolender, M., and Bak, S. Verification challenges in f-16 ground collision avoidance and other automated maneuvers. In *ARCH@ ADHS*, pp. 208–217, 2018.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, 2008.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pp. 89–94. Presses universitaires de Louvain, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Page, E. S. Continuous inspection schemes. *Biometrika*, 41 (1/2):100–115, 1954.

Rotman, N. H., Schapira, M., and Tamar, A. Online safety assurance for deep reinforcement learning. *arXiv preprint arXiv:2010.03625*, 2020.

Rousseeuw, P. J. and Leroy, A. M. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

Sadik, S. and Gruenwald, L. Research issues in outlier detection for data streams. *Acm Sigkdd Explorations Newsletter*, 15(1):33–40, 2014.

Sage, H., De Mathelin, M., and Ostertag, E. Robust control of robot manipulators: a survey. *International Journal of control*, 72(16):1498–1522, 1999.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pp. 146–157. Springer, 2017.

Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., Platt, J. C., et al. Support vector method for novelty detection. In *NIPS*, volume 12, pp. 582–588. Citeseer, 1999.

Thudumu, S., Branch, P., Jin, J., and Singh, J. J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):1–30, 2020.

Zhai, S., Cheng, Y., Lu, W., and Zhang, Z. Deep structured energy based models for anomaly detection. In *International Conference on Machine Learning*, pp. 1100–1109. PMLR, 2016.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

Appendix

A. Related Works

Classical anomaly detection algorithms can be categorized into one of the following groups: statistics-based (Rousseeuw & Leroy, 2005), classification-based (Liu et al., 2008), and distance-based (Breunig et al., 2000). Our work is closest to the distance-based methods. Such methods use the distance of each data point to its nearest neighbor data point or data cluster in order to evaluate how anomalous the given data point is. This group of methods relies on having the appropriate distance criterion for the data. Nevertheless, such algorithms experience problems due to the exponential computational growth in high-dimensional space and can not maintain their efficiency and performance. This is because the computational complexity of the above algorithms is directly related to the increase in data size and the number of samples. Increasing the dimensions leads to scattered cases and makes them less dense (Thudumu et al., 2020). Aggarwal (2017) suggested that almost any technique based primarily on the concept of proximity is qualitatively degraded in a high-dimensional space and should therefore be defined in a more meaningful way. Although Manhattan and Euclidean distance metrics are among the most used approaches for distance-based anomaly detection, they suffer in high dimensional cases (Sadik & Gruenwald, 2014). One may address the high dimensionality in data by projecting the dataset into a lower dimension space. It can be done in various ways; for example, principal component analysis or Laplacian eigenmaps are two different approaches to do so (Ham et al., 2004).

In addition to classical anomaly detection methods, deep learning-based methods have also gained some attention. These methods can be broadly categorized into: unsupervised, semi-supervised, hybrid (Erfani et al., 2016), and one class learners (Chalapathy et al., 2018; Schölkopf et al., 1999). Deep networks are capable of handling high-dimensional data properly. Autoencoders (unsupervised) learn to reconstruct the nominal data; hence, when given an outlier data point, it ends up having a high reconstruction loss because the anomalous data is taken from another distribution most likely (An & Cho, 2015). In this case, the reconstruction loss acts as the anomaly signal. Besides autoencoders, Schlegl et al. (2017), Zhai et al. (2016), and Zong et al. (2018) use energy-based models, deep auto-encoding Gaussian mixture models, and GANs to detect anomalies, respectively. In addition to these approaches, recurrent neural network (RNN) based methods have flourished in anomaly detection regarding sequential data. One can use RNNs to learn about the nominal time series data in a supervised fashion, then use the learned network as the anomaly detector. Malhotra et al. (2015) proposed to have stacked LSTM-based prediction model. Then, anomalies are detected using the prediction error distribution. Chauhan & Vig (2015) used a similar approach to detect various types of anomaly injected existing in the data.

Closest to our approach is Rotman et al. (2020)'s method. They propose a method using which the agent can switch from the learned policy to a safe, heuristic-based policy whenever it realizes that decisions no longer seem reasonable. It is done by measuring uncertainties regarding the observed states, the policy, and the value function. Although their goal is similar to ours, in our method, we use predictive knowledge as a tool to measure the level of uncertainty rather than having an ensemble of policies and storing the history of observed data. We also show the results of our method on more RL-friendly test bases, like OpenAI Gym and Bullet physics.

Our approach uses a distance-based method to calculate the anomaly scores while utilizing an RNN-based predictor (RIQN) to take advantage of the underlying correlation in the high-dimensional dataset. The scalability of the RIQN also helps with tackling the curse of dimensionality. Once the autoregressive RIQN has provided the prediction for features at a particular time step, the distance between the true data point and the estimated distribution can be measured. Ideally, for nominal data points, one expects this distance to be small and for anomalous data points to be large, similar to the reconstruction loss in autoencoders. We discuss this further in detail in Section 3. Overall, to the best of our knowledge, there is no work on detecting anomalies injected into an environment's transition dynamics while the agent is able to perform as well as before. Thus, our proposed method is the first to give RL agents such capability.

B. Control Systems Analysis Framework

Control Systems Analysis Framework (CSAF) serves as an advanced model to develop offline and online control assurance schemes. Roughly speaking, the dynamics are nonlinear, have about 10-20 dimensions (continuous state variables), and are hybrid in the sense of discontinuous ODEs, but not with jumps in the state (Heidlauf et al., 2018). Regarding the policy, CSAF comes with pre-defined finite-state-based policies that can perform reasonably well in the simulation. However, once the anomalies are injected, policies fail to complete the given tasks perfectly. Figure 2 (a) shows a simple demonstration of main components in the F16 system, and (b) demonstrates what a sample state looks like when it is rendered. The nominal

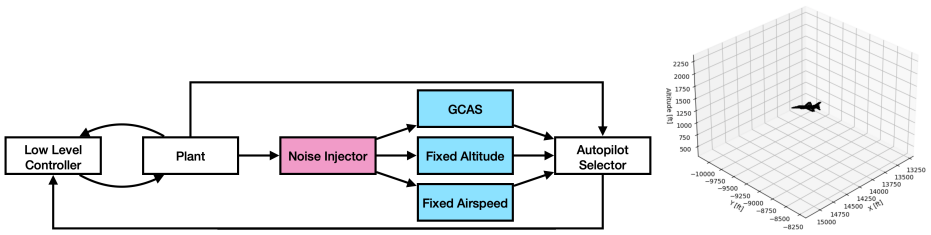

Figure 2. An overall view into the F16 system. (a) The blue components are autopilots, or policies. The noise injector component, shown in pink, does not originally exist in the system. We have added it manually to study the effect of noise. (b) Shows a rendered state from the F16 simulator.

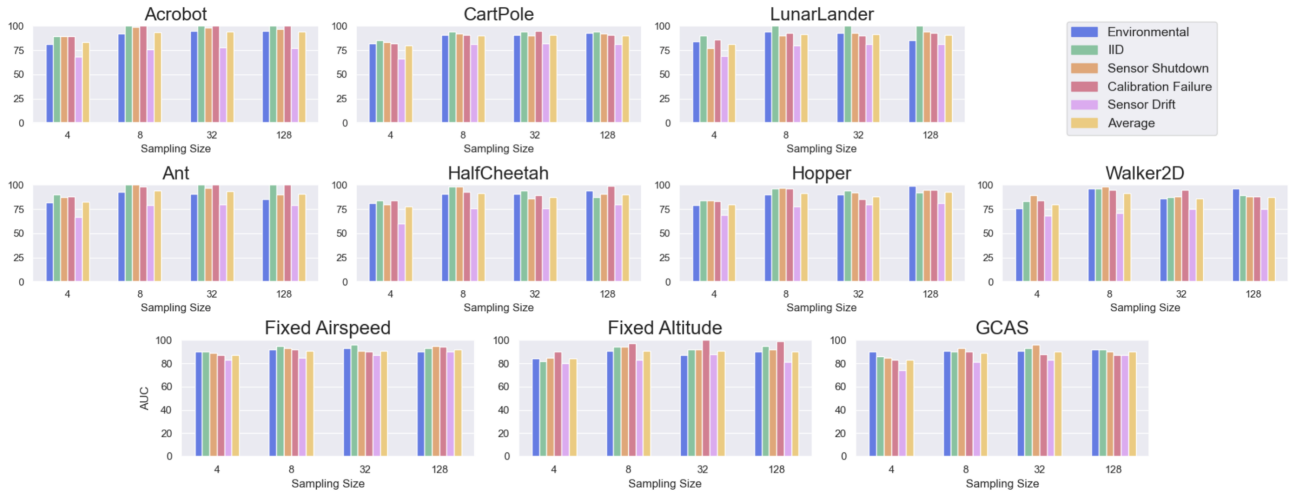

Figure 3. The impact of sampling size on anomaly detection per each type of anomalies. Horizon is set to 1.

system (by removing the noise injector component) works in the following fashion: initially, the plant outputs the initial state to autopilots and the autopilot selector. Each autopilot depending on its purpose, gives an action to the autopilot selector. The autopilot selector chooses which action needs to be taken, thus returning that to the low-level controller (LLC). Since plant and LLC have higher frequencies than autopilots, they communicate for a number of times until it is time for autopilots to take another action based on the state given by the plant. The noise injector that sits right after the plant adds noise to the state in order to inject intended anomalies into the system. The noise injector can simulate all types of anomalies described in Section 4. For the purpose of anomaly detection, we work with each autopilot one at a time to study how anomalies affect the system with respect to each autopilot. For example, we remove fixed altitude and GCAS components to analyze our method based on the fixed airspeed autopilot. Similarly, we do the same for the other autopilots. CSAF is under active development. It will become open-source in the near future under MIT license.

C. More Experiments

C.1. Horizon

Figure 4 demonstrates more in detail information for each anomaly type in the CSAF F16 setting. These results are very much aligned with the ones presented in the main paper, Section 4.

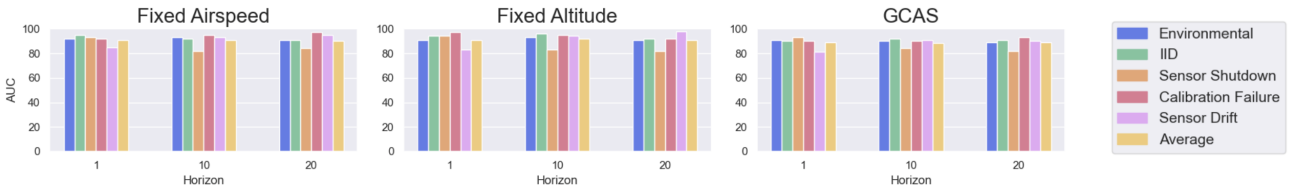

Figure 4. How changing the horizon affects anomaly detection in terms of AUC in F16.

Table 3. Comparison of AUCs obtained from RIQN and N-RIQN models. Results are shown for three environments each taken from a different setting.

Environment	$\Delta = 1$		$\Delta = 10$		$\Delta = 20$	
	RIQN	N-RIQN	RIQN	N-RIQN	RIQN	N-RIQN
Acrobot	95.1	81	93.2	78	93	79
Ant	94.4	83	96	81	94.8	80
Fixed Airspeed	92.3	80	93.3	80	93	81

C.2. Sampling size

The next parameter to investigate is the sampling size, s. It basically determines with how many samples we should estimate the true value of a feature. Naturally, one might think the more samples, the better. However, it comes with a larger prediction tree, thus more processing time. Also, a distribution could be represented with a handful of samples. Thus there would be no need to take a lot of samples. On the other hand, decreasing the number of samples results in a bad estimation and inaccurate predictions. As shown in Figure 3, having $s = 4$ underperforms the case with $s = 8, 32, 128$ in all environments. Also, increasing the sampling size to more than 8 gives no absolute benefit while adding processing overhead. This suggests that the true underlying distribution can be almost accurately estimated using 8 samples. Therefore, $s = 8$ is considered to be the best choice. It solves the issues of under-sampling for prediction and processing time.

C.3. Role of memory

To answer the fourth question raised in Section 4, we have removed the memory module from the RIQN and trained it over the same dataset as the original RIQN anomaly detector. One major issue raised during the N-RIQN anomaly detection was the accumulation of loss, which was solved using Scheduled Sampling during its training. However, it still underperforms the RIQN in anomaly detection. This becomes noticeable, especially when the horizon gets longer. Table 3 compares the average AUC obtained using RIQN and N-RIQN models over three environments: Acrobot, Ant, Fixed Airspeed. Therefore, having the memory in the model seems crucial to have an accurate prediction, leading to high AUCs.

C.4. Delay in detection

Table 4 includes the delay in anomaly detection using two different signals: the anomaly scores calculated based on our method, and the original features. As it can be seen, our method has a significantly lower delay in detecting anomalies.

C.5. Epistemic uncertainty

To study the effect of having an ensemble of models instead of just one model, we experimented an ensemble of five models versus only one model. According to our experiments, we can see that the ensemble perform better regarding the obtained AUC compared to only one model, as presented in Table 5. The reason for such an improvement is because with an ensemble of models, it is easier to address the epistemic uncertainty.

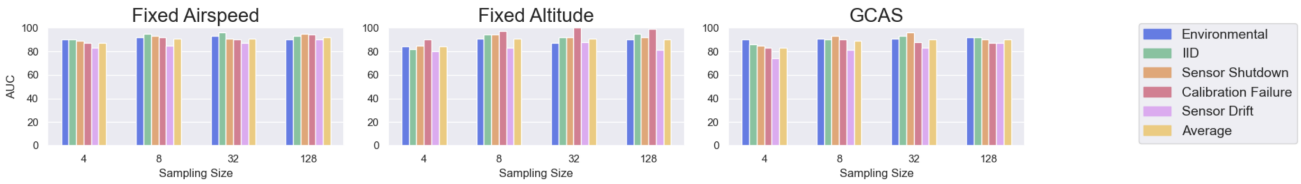

Figure 5. The impact of sampling size on anomaly detection per each type of anomalies in CSAF F16 case studies. Horizon is set to 1.

Table 4. Anomaly detection delay using anomaly scores vs. features. Sampling size is set to 8.

Environment	$\Delta = 1$		$\Delta = 10$		$\Delta = 20$	
	Anomaly scores	Features	Anomaly scores	Features	Anomaly scores	Features
Acrobot	2.5	6.4	11.4	17.1	22	28.1
CartPole	2.3	7.5	11	19	23	28
LunarLander	3.3	7	12	17.5	24.5	30.1
Ant	2.1	9	11	17	22.2	26.4
HalfCheetah	2.3	7.3	11.1	18.5	23.3	29.3
Hopper	2	5.8	11	18	20.9	28.1
Walker2D	2	9.3	11.1	18	22.5	28.7
Fixed Airspeed	4.2	9.5	15.1	20.5	27.1	33.8
Fixed Altitude	3.7	9	14	21.1	26	32
GCAS	4.5	9.5	13.3	21	24.1	29.6

D. Details on RIQN

The overall RIQN's architecture is shown in Figure 6. It has 64 neurons in the first fully connected layer, 64 memory cells in its memory which is a GRU model (Cho et al., 2014), and 64 neurons in both the second and the third fully connected layers. Between the first fully connected layer, the GRU, and the second fully connected layer there is a skip connection. The skip connection helps to improve the performance of the network. During the training, we use Adam optimizer with a learning rate of either 0.001 or 0.01.

One concern specific to the autoregressive RIQN is how to utilize the distributional estimations for long horizons ($\Delta > 1$). For $\Delta > 1$, eventually, the number of samples of feature estimations will exponentially grow. For example, if we take s samples for each feature to represent the estimated distribution, ultimately, we will have $M * s^H$ samples representing $\hat{f}_{t+\Delta}$, where M is the size of the feature vector. To overcome this issue, we calculate the mean of samples from time steps $t + 2$ until $t + \Delta$, demonstrated in Figure 7 (a) and (b). Using this approach, while we keep taking advantage of the distributional returns by the predictor, we are able to control how big our prediction tree will be, no matter what the horizon is. Doing so offers a number of benefits: first, it prevents the explosion of samples at $t + \Delta$ by keeping the sampling size fixed: $M * s$. Second, it helps to mitigate the problem of accumulation of loss. And third, we can easily control the size of the prediction tree based on the computation power and sampling space by changing the time to start taking means.

Table 5. AUC results with $\Delta = 1, 10$, to study the effect of ensemble size. #M refers to the number of models in the ensemble.

Environment	$\Delta = 1$		$\Delta = 10$	
	#M=1	#M=5	#M=1	#M=5
Acrobot	93.4	95.1	93.2	93.2
CartPole	89.8	91	93.4	93.9
LunarLander	91.4	91	91.4	91.8
Ant	94	94.4	95.8	96
HalfCheetah	91.2	92	90.6	91.6
Hopper	91.4	91.6	90.6	91
Walker2D	91.2	92.2	89	90.5
Fixed Airspeed	91.8	92.3	93	93.3
Fixed Altitude	91	91.1	91.2	93.2
GCAS	90.2	91	90	90.4

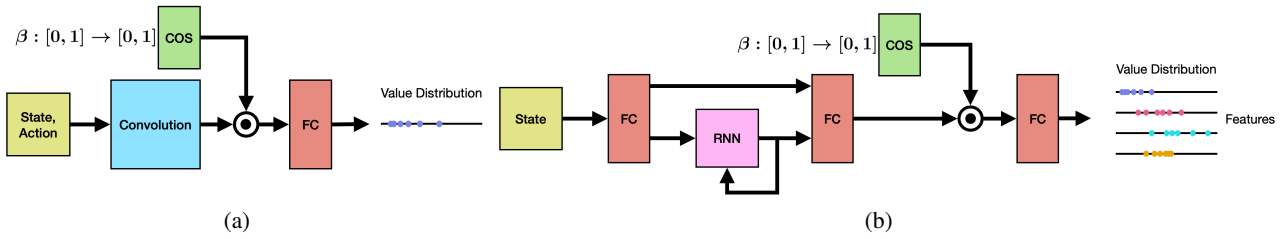

Figure 6. Comparing (a) IQN and (b) RIQ network architectures.

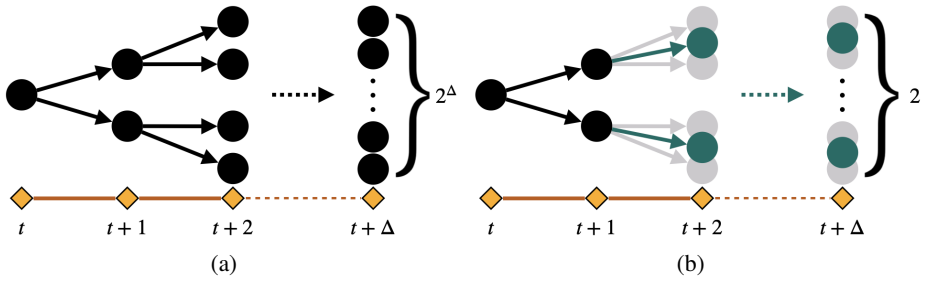

Figure 7. How estimation samples exponentially grow at Δ for **one** feature. (b) shows the case where at each time step, the prediction is sampled using 2 values. Eventually at $t + \Delta$, we will have $1 * 2^{\Delta}$. (c) shows the case where starting from $t + 2$, we calculate the mean of taken samples and continue with them rather than all the samples. From $t + 2$, samples are shown in grey, and their corresponding means are shown in teal. This way, at $t + \Delta$, we will have $1 * 2$ samples.