Memory-Based Distribution Shift Detection for Learning Enabled Cyber-Physical Systems with Statistical Guarantees

YAHAN YANG, RAMNEET KAUR, SOURADEEP DUTTA, and INSUP LEE, University of Pennsylvania, Pennsylvania, USA

Incorporating learning based components in the current state-of-the-art cyber-physical systems (CPS) has been a challenge due to the brittleness of the underlying deep neural networks. On the bright side, if executed correctly with safety guarantees, this has the ability to revolutionize domains like autonomous systems, medicine, and other safety-critical domains. This is because it would allow system designers to use high-dimensional outputs from sensors like camera and LiDAR. The trepidation in deploying systems with vision and LiDAR components comes from incidents of catastrophic failures in the real world. Recent reports of self-driving cars running into difficult to handle scenarios is ingrained in the software components which handle such sensor inputs.

The ability to handle such high-dimensional signals is due to the explosion of algorithms which use deep neural networks. Sadly, the reason behind the safety issues is also due to deep neural networks themselves. The pitfalls occur due to possible over-fitting and lack of awareness about the blind spots induced by the training distribution. Ideally, system designers would wish to cover as many scenarios during training as possible. However, achieving a meaningful coverage is impossible. This naturally leads to the following question: is it feasible to flag out-of-distribution (OOD) samples without causing too many false alarms? Such an OOD detector should be executable in a fashion that is computationally efficient. This is because OOD detectors often are executed as frequently as the sensors are sampled.

Our aim in this article is to build an effective anomaly detector. To this end, we propose the idea of a memory bank to cache data samples which are representative enough to cover most of the in-distribution data. The similarity with respect to such samples can be a measure of familiarity of the test input. This is made possible by an appropriate choice of distance function tailored to the type of sensor we are interested in. Additionally, we adapt conformal anomaly detection framework to capture the distribution shifts with a guarantee of false alarm rate. We report the performance of our technique on two challenging scenarios: a self-driving car setting implemented inside the simulator CARLA with image inputs and autonomous racing car navigation setting with LiDAR inputs. From the experiments, it is clear that a deviation from the in-distribution setting can potentially lead to unsafe behavior. It should be noted that not all OOD inputs lead to precarious situations in practice, but staying in-distribution is akin to staying within a safety bubble and predictable behavior. An added benefit of our memory-based approach is that the OOD detector produces interpretable feedback for a human designer. This is of utmost importance since it recommends a potential

This work was supported in part by ARO W911NF-20-1-0080, NSF-1915398, NSF-2125561, and SRC Task 2894.001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Army Research Office (ARO), the Department of Defense, the National Science Foundation (NSF), or the United States Government.

Authors' address: Y. Yang, R. Kaur, S. Dutta, and I. Lee, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104, USA; e-mails: yangy96@seas.upenn.edu, ramneetk@seas.upenn.edu, duttaso@seas.upenn.edu, lee@seas.upenn.edu.

ACM 2378-962X/2024/05-ART21
https://doi.org/10.1145/3643892

fix for the situation as well. In other competing approaches, such feedback is difficult to obtain due to reliance on techniques which use variational autoencoders.

CCS Concepts: • **Computer systems organization → Robotic autonomy;**

Additional Key Words and Phrases: Conformal prediction, autonomous driving, OOD detection

ACM Reference Format:
Yahan Yang, Ramneet Kaur, Souradeep Dutta, and Insup Lee. 2024. Memory-Based Distribution Shift Detection for Learning Enabled Cyber-Physical Systems with Statistical Guarantees. *ACM Trans. Cyber-Phys. Syst.* 8, 2, Article 21 (May 2024), 28 pages. https://doi.org/10.1145/3643892

1 INTRODUCTION

In the past few years, tremendous progress has been made toward improving the building blocks of fully autonomous systems. Components like LiDAR and camera, which were out of reach for most day-to-day applications, have become an integral part of the sensor array for driverless cars. Yet, there is a missing component that limits the adoption of autonomous systems from a well-monitored lab setting to the open world. The missing piece is safety. The consensus right now is to make cautious incremental progress before full autonomy is reached [45]. The first step is to incorporate **Learning-Enabled Components (LECs)** in modern **Cyber-Physical Systems (CPS)** in a limited capacity. Such LECs often rely on having access to a large corpus of data designed by humans. This often relies on well-trained human experts to build such a collection. Although often a big challenge in practice, we assume in this article that such a dataset is available during the training phase.

We direct the attention of the reader to one such LEC in a modern car [12]. A very humble setting compared to a driverless car is a simple lane keeping controller. The control algorithm receives a sensor feed in the form of a video and makes slight adjustments to keep the vehicle on the road. This happens under the supervision of a human driver who is still responsible for the safety of the vehicle. Contrast this scenario with that of a fully self-driving car equipped with an **Automated Driving System (ADS)**. The software in this case has full control of the car at multiple levels of abstraction, starting from navigation and choice of route to maneuvering the vehicle in a heavy traffic scenario. If realized correctly, this can have deep ramifications. It can lead to reduction of accidents and improve general vehicle safety. In addition, it can lead to higher mobility for seniors and those who cannot drive. For this work, we use such a lane keeping controller based on video input.

The workhorse of machine learning systems are **Deep Neural Networks (DNNs)**. These high-capacity function approximators can take inputs from a camera and LiDAR and produce a desired output. To function properly, DNNs go through a process of training to correct their mistakes by learning from a large corpus of curated data. Training large-scale DNNs are possible only due to the development of well-engineered gradient descent tools such as PyTorch [44] and TensorFlow [4], as well as improvements in specialized hardware like GPUs. It is not surprising anymore for DNNs to be able to perform superhuman performance in Atari games [42] and Alpha-Go [51]. But this does not come without its potential pitfalls. The mode of functioning of a DNN is quite different from the way humans operate. There is a clear lack of similarity when it comes to understanding an image. Concepts like persistency of objects, segmentation, and depth of vision do not happen naturally in a typical end-to-end training scenario. The ability to classify objects occurs by taking the right steps to *fit* a high-capacity function to the training data by controlling its labeling error rate. DNNs in the most basic form have millions of nodes and parameters, which are arranged in a fashion so as to form a directed acyclic computational graph. This complex nature of the

graph along with its size makes it particularly difficult to be analyzable by human experts. This means that computer vision systems are more likely to make mistakes in ways that humans are not. A well-known example of such a phenomenon is the presence of adversarial perturbations to an otherwise clean image. This is referred to as an adversarial attack in the literature. Here, one makes imperceptible changes to an image which are completely ignored by human vision but can mislead the DNN. This can affect the safety of autonomous systems adversely. For instance, it is fairly straightforward to alter a speed limit sign to be interpreted as a different speed or change a stop sign to a speed limit sign.

If it were true that neural networks can automatically learn concepts like humans, then it would generalize well outside the training region, implying lesser concern for the CPS designer. But it is well known that, when pushed outside of their training region, neural networks can perform in an unpredictable fashion. Statistical machine learning tools can compute upper bounds on the generalization errors of a learned model. But these are often overly conservative, due to the degree of over-parameterization in a DNN.

Out-of-Distribution (OOD) detection [24, 32, 41] has been the focus of attention in a large section of the literature. This is mainly because of the promise it offers for infusing robustness in an otherwise delicate workflow of the DNNs. The main target of the research community working on these approaches is to analyze the robustness of the system when exposed to unknown settings. But it is apparent from the literature that such approaches do not address the CPS settings which function in the closed loop producing closely related samples but is more geared toward standard classification datasets like MNIST and CIFAR10. The main contribution of this article is that it offers an avenue to detect anomalous inputs in real time and in an interpretable fashion. The overall workflow involves building a representation system for familiar inputs (i.e., **in-Distribution (iD)** data as it is commonly called) while minimizing the number of such representative samples one uses in the process. Such prototype samples are referred to as *memories* in our work. To discover these prototypes, we use well-established computer vision techniques to compute the similarity of two images.

To rigorously control false detection by the proposed approach, we leverage a conformal anomaly detection framework [37]. This framework aims to test if an input conforms to the training distribution by assigning a non-conformity score to the input. The higher the score, the more non-conforming the input is to the training distribution. The probability of false detection by conformal anomaly detection is upper bounded by the detection threshold. The detection performance, however, depends on the choice of the **Non-Conformity Measure (NCM)** used as described in the original framework [6]. We propose using the **Statistical Similarity Index Metric (SSIM)** in the conformal anomaly detection framework for OOD detection in CPS. With inputs coming in a time-series sequence to the CPS, we propose to use the **Harmonic Mean p-value (HMP)** method [63] to combine conformal predictions from individual inputs (in the sequence) for detection on the sequence. The HMP method preserves the false detection guarantees from the conformal anomaly detection framework without assuming that the conformal predictions are computed independently [63].

Another advantage of the proposed framework is its ability to produce interpretable outcomes. This is accomplished using samples from the training dataset. In explainable machine learning systems, comparative methods which relate a test sample to a witness from the dataset [13, 52] are fairly well accepted. It is important to note that such feedback is often useful to the system designer especially. Such feedback can often be turned into a potential fix for the improper behavior, as will be evident later on in the article. In this work, we consider two broad sets of case studies, one of which involves a camera feed in the form of a video, and the other being in the form of LiDAR inputs. The video inputs feed into an **Advanced Emergency Braking System**

(AEBS) and an end-to-end self driving system presented in the work of Cai and Koutsoukos [11]. The system is subjected to varying types of distribution shifts, such as a shift from the training weather (low precipitation), lighting conditions (day), leading obstacles (car), and clean (or non-adversarial) images. The other setting is that of LiDAR inputs. The anomalous inputs appear as random reflections encountered by the emitted light beam. These reflections push the network outside of its trusted zone, causing a deviation from safe behavior. Our experimental results show that our algorithm is able to achieve state-of-the-art results of distribution shift detection in an effective and interpretable way. We summarize our contributions next.

Contributions. First, we propose an algorithm to build a memory system composed of representative data points to capture the distribution and assign non-conformity scores. Second, we demonstrate how these scores can be combined to detect distribution shifts using the HMP and the **Inductive Conformal Anomaly Detection (ICAD)** framework, thereby exploiting the statistical guarantees that come with it. Third, we propose a way to compute a feedback image, which can help the system designer with an interpretable report for the OOD input.

***Improvements Compared to Our Previous Work* [66].** The main improvement that we propose in this article is a statistical guarantee on the **False Detection Rate (FDR)** for window-based OOD detection. This utilizes the backbone of the memory system to compute the distance between a test sample and stored memories, employing it as a non-conformity score within the ICAD framework. Image and LiDAR frames obtained from a continuous stream of sensor inputs are often correlated in practice, which violates the independence assumption typically used. To deal with this issue, we use the HMP method to combine p-values in a window and to upper bound the FDR on windows. This moves the results from an image-level detection to guarantees on the windows. Our experimental results show that we can effectively control the **False-Positive (FP)** detection rate with the statistical guarantee. Additionally, we include a systematic analysis of the effect of different window sizes and different expected FDR for trace-wise detection of OOD samples. We compare these results with existing state-of-the-art **Variational Autoencoder (VAE)**-based OOD detectors [11, 54]. The details are discussed in Section 6 and Section 7.

2 RELATED WORK

Autonomous systems broadly categorize a class of systems where certain tasks can be performed with minimal human intervention. Such behaviors are often hard to encode on a case-by-case basis. This necessitates learning-based components like DNNs being placed in the workflow. However, the use of DNNs has produced safety concerns. This has piqued an interest from several domains, which have contributed in producing varying levels of assurance cases [29]. This has ranged from verification of DNNs against some specifications to runtime techniques for detection and recovery. Synthesizing deep learning systems which are robust by construction [17, 46] is often challenging in the more general setting. Yet, if it is the case that the property of correctness can be captured more precisely as a set propagation problem, then more hard guarantees can be given. In the literature, this is referred to as the domain of neural network verification [21, 23, 27, 28, 64, 65]. A more detailed survey of verification approaches can be found in the work of Sankaranarayanan et al. [49]. Despite serious efforts, it was quickly apparent that it is often difficult to verify applications which use images as inputs, not just because of the computational aspects but because it is hard to specify the problem itself. Defining what verification would mean in the context of images is a challenge. Recently, there has been interest in addressing this issue through an approach of evidence-based trust, such as predicting the performance of these networks in novel scenarios for deploying it in real-world systems [33]. In this article, we also follow the evidence-based trust approach. If the system is able to justify its decision by presenting evidence, we deem that as

more likely to be the correct behavior. Next, we provide a summary of techniques specialized to images.

OOD detection has been extensively studied in classification problem settings for stand-alone LECs [24, 30, 31, 34, 41, 56, 67]. These approaches either use differences in the geometrical or statistical properties of the iD and OOD data for detecting a shift in the model behavior. OOD detection through safety envelopes, in CPS with low-dimensional input space sensors such as GPS, has been studied in the past [57]. Recently, there has been growing interest for detection of OOD and adversarial inputs in closed-loop CPS using high-dimensional sensors like a camera [11, 18, 35, 36, 47, 53, 54].

Sundar et al. [54] propose using KL divergence in the latent space of β-VAE for detection of individual images as OOD. Feng et al. [18] propose using KL divergence in the horizontal and vertical latent sub-space of the 3D convolutional VAE from the specified prior for detection of OOD traces. The input to 3D convolutional VAE is a sequence of frames (or the trace to be detected). To our knowledge, such techniques do not provide FDR guarantees on OOD detection.

ICAD has recently been utilized for controlling the false OOD detection on iD data [8, 11, 30, 36, 47]. iDECODe [30] proposes using error in the equivariant behavior of a model as the non-conformity score in CAD for OOD detection. The equivariance is learned on the data drawn from the training distribution and with respect to a set of transformations such as rotation on images. Bates et al. [8] propose combining conformal predictions from different channels and layers of convolutional neural networks for OOD detection. Whereas Bates et al. Bates et al. [8] and Kaur et al. [30] focus on detection of individual datapoints as OOD, we consider the problem of OOD detection in time-series data to CPS.

Cai and Koutsoukos [11], Ramakrishna et al. [47], and Kaur et al. [36] leverage ICAD for OOD detection on a sequence of time-series inputs. Cai and Koutsoukos [11] propose using reconstruction error by VAE on the input image (or frame) as a non-conformity score in the ICAD framework [38] for detection of OOD frames. They further apply the Martingale test [60] along with the cumulative sum procedure (CUSUM) [7] with a window of the past and present predictions for robust detection of OOD traces.[1] Ramakrishna et al. [47] use KL divergence between the disentangled feature space of β-VAE and normal distribution as the non-conformity score in ICAD for OOD detection of a single frame. They use the Martingale test along with CUSUM for detecting OOD traces. CODiT [36] uses error in the temporal equivariance learned by a VAE model on the training distribution of time-series windows as the non-conformity score in ICAD for OOD detection in time-series data. We propose to use memory-based distance as the non-conformity score in ICAD for OOD detection in time-series data with interpretable explanations on the detection.

To the best of our knowledge, all existing approaches for OOD detection in CPS with LEC are tied to VAE. Either reconstruction error from VAE on the input image or KL divergence in the latent space of the VAE is used for OOD detection in these approaches. Training VAEs often requires careful manual tuning [5], and the quality of the training decides the efficacy of the downstream processes. Here we set ourselves apart by not having to depend on a well-functioning VAE. In addition, unlike our approach, none of the existing approaches except for that of Ramakrishna et al. [47] provides interpretability on the source of OOD-ness of the input. We show that our approach can be extended to the case of LiDAR inputs as well without any conceptual modification.

[1]Cai and Koutsoukos [11] also consider distance of the input from the center of the hypersphere learned by a support vector training data description as another score in ICAD for OOD detection in CPS with LEC.

(a) Car does not detect the biker, leading to a crash

(b) Frame detected as OOD; the region in red shows the pixels responsible for deviation

Fig. 1. Deviation from training data leads to a crash with a biker as the front object. Training data only had cars as front objects. Our proposed method could detect deviations from iD data for detecting such OODs.

3 MOTIVATION AND PROBLEM STATEMENT

The level of autonomy a CPS has to offer is often decided by how well a designer leverages the LECs. Detection of OOD is one of the ways we can safeguard systems from unwarranted behavior. In Figure 1(a), we show an example of a setting where the car is running an AEBS controller. The controller uses the system states and the video feeds from the camera to sense the positions of the closest leading object on the road. The controller's job is to automatically brake the car if it crosses a certain distance threshold from the leading vehicle. What we observe is that because during training the DNN experienced just cars, it never learned to react to bikes on the road. What happens next is that the DNN completely misjudges a bike in the video and the controller ends up with an accident. What we would like to propose here is a method to detect such a shift in distribution.

Problem Statement. We would like to solve the problem of being able to alarm the system about distribution shifts in real time with statistical guarantees. It is extremely challenging to sample high-dimensional inputs space in an exhaustive fashion. This would mean careful analysis of the training time iD data to come up with an effective detector that can act in real time. Additionally, it is desirable that such an alarm system produces interpretable behavior. It is often the case that DNNs, due to their black-box nature, do not offer an explanation to their decisions. Here, we would like to take up the challenge of being able to point to an explanation when samples are iD or OOD. We demonstrate this in Figure 1(b), in which the system not only flags the image with the biker ahead as OOD, but selects a set of pixels demarcating the biker to communicate why it decided to label it as an OOD.

4 BACKGROUND

In this section, we walk through some of the basic concepts and proof required for our approach.

4.1 Clustering with Medoids

One of the promising steps to build an understanding of the data distribution is through unsupervised clustering. In a fashion similar to *k-means* clustering, we wish to form partitions of the data into distinct groups or clusters. Clustering with k-means is a well-known tool but has its challenges when used in the context of images. An issue with k-means is that it can potentially produce virtual cluster centers which are absent in the original dataset. This is essentially because a simple mean of two (or more) images might not correspond to a real image. This is important

for us since we wish to use these centers to form interpretable predictions which can answer *why* something was flagged as OOD. The other issue with vanilla k-means is that it is often susceptible to outliers in the data. Hence, we restrict ourselves to partitioning around points which are present in the data.

The algorithm that achieves this is PAM [2], which is short for *Partitioning Around Medoids*. Intuitively, the algorithm tries to search for centrally located data samples called *medoids* and are used to define the cluster boundaries in a nearest medoid sense. Let us assume that the set $\mathcal{S}$ is equipped with a distance metric $\mathcal{D} : (s_1, s_2) \rightarrow \mathbb{R}$, for $s_i \in \mathcal{S}$ and $n = |\mathcal{S}|$. Given a dataset $\mathcal{S}$, PAM tries to select a set of r medoids - $M_r : \{m_1, m_2, \ldots, m_r\}$ such that the following cost is minimized:

$$Cost(M_r) = \sum_{i=1}^{n} \min_{m_j \in M_r} \mathcal{D}(m_j, s_i). \tag{1}$$

We assume that the inner minimization is always possible, and we are able to break ties arbitrarily among distinct members of the set $\mathcal{S}$.

Algorithms. The challenge with PAM is that the naive implementation has a runtime complexity of $O(n^2r^2)$ [50]. Even though there exist faster variants, it is still largely inaccessible for applications at the scale of image datasets generated from autonomous driving scenarios. To circumvent this challenge, we introduce a variant of the *Clustering Large Applications based upon Randomized Search* (CLARANS) [43] algorithm in Section 5.2. It combines randomized global search with a local clustering method. The medoids identified by minimizing the objective in Equation (1) are referred to as *memories* from here on.

4.2 Structural Similarity Index Metric

A fundamental challenge in dealing with images is to capture human perceptual similarity with a mathematically meaningful distance function. To the best of our knowledge, the right candidate for this purpose is SSIM. This was first introduced in the work of Wang et al. [62] and has gained widespread popularity. It computes the degree to which two images are similar to a human eye and was used to compute the degradation quality of an image. SSIM is designed to capture statistical similarity between images. Figure 2 gives an example of how SSIM successfully captures the perceptually difference of the images. To human eyes, the image on the left is more similar to the reference image (middle one) compared to the right one, which is reflected by the SSIM distances below. This makes our system more robust to random noise in comparison to vanilla DNNs. It has been used to capture image similarity for adversarial sticker attacks as well [39]. We exploit this feature in the context of videos, where subsequent frames are not worlds apart but are quite correlated in their information content.

We state the original SSIM distance function next. Assume we have two images $\mathsf{A}_1 \in \mathbb{R}^N$ and $\mathsf{A}_2 \in \mathbb{R}^N$. This allows us to compute three terms: a luminance distortion term, a contrast distortion term, and a correlation term as follows:

$$l(\mathsf{A}_1, \mathsf{A}_2) = \frac{2\bar{\mathsf{A}_1}\bar{\mathsf{A}_2} + c_1}{\bar{\mathsf{A}_1}^2 + \bar{\mathsf{A}_2}^2 + c_1}, \tag{2}$$

$$c(\mathsf{A}_1, \mathsf{A}_2) = \frac{2s_{\mathsf{A}_1}s_{\mathsf{A}_2} + c_2}{s_{\mathsf{A}_1}^2 + s_{\mathsf{A}_2}^2 + c_2}, \tag{3}$$

$$s(\mathsf{A}_1, \mathsf{A}_2) = \frac{s_{\mathsf{A}_1,\mathsf{A}_2} + c_3}{s_{\mathsf{A}_1}s_{\mathsf{A}_2} + c_3}, \tag{4}$$

Fig. 2. SSIM distance illustration. Here we show an example of how SSIM distance works.

where $\bar{A_1}, \bar{A_2}, s^2_{A_1}, s^2_{A_2}$, and s_{A_1,A_2} are the local mean, local variance, and local covariance between A_1 and A_2. The scalar terms c_1, c_2, c_3 aim to capture the saturation effects of the visual system and provide numerical stability. The terms computed above capture the local difference in some chosen window in the image. The combination across all such local windows gives the SSIM. With $c_3 = c_2/2$, SSIM can be written in the following form:

$$\begin{aligned} SSIM(A_1, A_2) &= S_1(A_1, A_2)S_2(A_1, A_2) \\ S_1(A_1, A_2) &= l(A_1, A_2) \\ S_2(A_1, A_2) &= c(A_1, A_2)s(A_1, A_2). \end{aligned} \tag{5}$$

The computational structure of SSIM allows us to efficiently implement it in tools like PyTorch and accelerated using a GPU. This permits a scalable and efficient implementation inside our OOD detection framework. A large gamut of algorithms in Euclidean spaces evolved with the assumption of a true distance metric being present. To leverage these methods, it is important that we work with a distance function that is a true metric. The downside of SSIM is that it does not have the mathematical properties to be a true distance *metric*. But with some modifications, it can be turned into one. The details of this modification and the associated proof can be found in the work of Brunet et al. [10]. We use the modified SSIM to define a distance metric $\mathcal{D}(A_1, A_2)$ in this article. The use of a proper distance metric for images allows us to capture distribution shifts in a more meaningful way.

4.3 Inductive Conformal Prediction and ICAD

Conformal prediction [6] is a general framework for testing conformance of an input with respect to the training distribution. Conformance with the training distribution is quantitatively measured by an NCM, which is a real-valued function that assigns a non-conformity score α to the input with respect to data drawn from the training distribution. The higher the score is, the more non-conforming the input is with respect to the training data $X = \{x_1, x_2, \ldots, x_l\}$. NCMs based on nearest neighbors [59], support vector machines [59], random forests [14], and VAEs [11] have been proposed in the past.

Conformal anomaly detection [37] uses the non-conformity score from the conformal prediction framework to detect anomalous inputs. A p-value of the input x is computed by comparing its non-conformity score α_x with the scores of the training data:

$$p_{l+1} = \frac{|\{i = 1, \ldots, l : \alpha_x \leq \alpha_i\}| + 1}{l + 1}.$$

Here, $\{i = 1, \ldots, l : \alpha_i\}$ is the set of non-conformity scores computed for the training data from an NCM defined on the new set composed of the training data and the input x_{l+1}. If x_{l+1} is drawn from the training distribution, then its score is expected to lie within the range of the scores of the

training data and therefore higher p-values for the iD datapoints. Conformal anomaly detection detects an input as anomalous of its p-value lies below a specified detection threshold $\epsilon_{icad} \in (0, 1)$.

Recomputing scores for the training data for every new input might be computationally expensive (and even infeasible in real time) if computing the NCM is inefficient. ICAD [38] was proposed to resolve this issue. ICAD is based on the inductive version of the conformal prediction framework, where training data are divided into a proper training set $X_{tr} = \{x_1, \ldots, x_m\}$ and a calibration set $X_{cal} = \{x_{m+1}, \ldots, x_l\}$. NCM is defined on the proper training set and the p-value of the input x is computed by comparing its score with these scores of the calibration datapoints:

$$p\text{-}value(x) = \frac{|\{i = m+1, \ldots, l : \ \alpha_x \leq \alpha_i\}| + 1}{l - m + 1}. \tag{6}$$

The non-conformity scores of the calibration set are computed in the offline settings and used at the inference time to compute the p-value of an input. Again, the input detected as an anomaly if its p-value lies below a specified detection threshold $\epsilon_{icad} \in (0, 1)$.

Lemma 4.1 ([6]). *If an input x and the calibration datapoints are independent and identically distributed, then the p-value of x computed from (6) is uniformly distributed. The probability of misdetecting x as anomalous is therefore upper bounded by the detection threshold ϵ_{icad}.*

4.4 Combining p-Values Using HMP

Previous work [36] uses Fisher's method [20] to combine multiple p-values from the ICAD framework. To preserve FP (or detection) rate guarantees from the conformal prediction framework, Fisher's method requires the individual p-values to be independent [58]. Datapoints in a time-series window have a temporal dependency. So, we propose to use the HMP method [63] to combine p-values of the temporally dependent datapoints in a time-series window. We consider the following problem:

Given a window $\mathcal{W}$ of consecutive time-series datapoints $(x_1, x_2, \ldots, x_n)$, label $\mathcal{W}$ as iD or OOD. We pose this problem as a statistical hypothesis testing problem with the null hypothesis of $\mathcal{W} \sim$ iD. With the single hypothesis testing of $\mathcal{W} \sim$ iD, we propose a solution based on multiple testing of the single hypothesis. The p-value from each test in the multiple testing framework can be combined to test for the single or global null via averaging methods [61]. The HMP method [63] is one such method that can be used to combine the dependent p-values from multiple tests while testing for the global null. We compute n p-values $(p_1, p_2, \ldots, p_n)$ by performing a test on each datapoint in $\mathcal{W}$ and then combine these p-values from multiple tests by using the HMP method for testing the same global null of $\mathcal{W} \sim$ iD. We denote the combined p-value by p_{agg}:

$$p_{agg} = HMP(p_1, p_2, \ldots p_n) = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} w_i / p_i} \text{where } \sum_{i=1}^{n} w_i = 1, \tag{7}$$

where n is the total number of datapoints in the window. In this article, we assign equal weights to all p-values: $w_1 = w_2 = \cdots = w_n = 1/n$.

Let us denote hypotheses in the multiple hypothesis testing framework by $H_{0,1}, H_{0,2}, \ldots$. In the proposed solution, $H_{0,1} = H_{0,2} = \cdots = H_{0,n} = H_0$ of $\mathcal{W} \sim$ iD. $H_{0,1} = H_{0,2} = \cdots = H_{0,n}$ are tested by computing n p-values: $p_1, p_2, \ldots, p_n$ on the n datapoints in $\mathcal{W}$. We want to control the FP rate on the global null of $H_0 = \mathcal{W} \sim$ iD.

Theorem 1. *When all tests for the same null H_0 in multiple hypothesis testing are combined using the HMP method, the probability of incorrectly rejecting H_0 (FP rate) is upper bounded by the significance level ϵ. If the p_{agg} from the HMP method (7) is less than the significance level ϵ, then the*

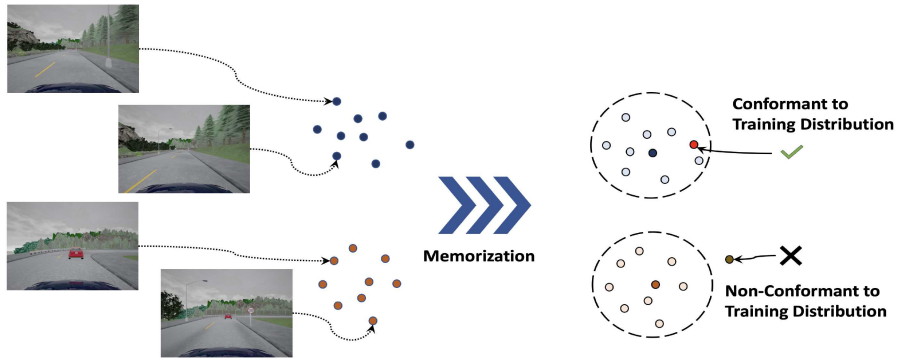


Fig. 3. Our approach can be summarized as follows. The memorization phase of the algorithm picks prototypical samples as memories. At runtime, the algorithm computes the distance of the input from its closest memory and uses it as the non-conformity score in the ICAD framework to determine the input's anomalous behavior with respect to the training distribution.

$H_0 = \mathcal{W} \sim iD$ is rejected. The probability of falsely rejecting H_0 by p_{agg} is therefore upper bounded by ϵ: $\mathbb{P}(p_{agg} \leq \epsilon) \leq \epsilon$.

Proof. The HMP method controls the strong-sense **Family-Wise Error Rate (FWER)** in multiple hypothesis testing (Results in the work of Wilson [63]). Since strong-sense FWER control implies the weak-sense FWER control, the HMP method controls the weak-sense FWER in multiple hypothesis testing.

Controlling weak-sense FWER at the significance level ϵ by combining p-values in multiple hypothesis testing for the same null H_0 is equivalent to controlling the FP rate while testing for the single hypothesis H_0 (Roquain [48, Remark 1.6]). □

The HMP method compares p_{agg} with a critical value ξ for detection ($p_{agg} < \xi$) [63]. For the desired FP rate ϵ, ξ can be calculated by inverting the following equation:

$$\epsilon = \int_{1/\xi}^{\infty} f_{Laudau}\left(x|log(n) + 0.874, \frac{\pi}{2}\right) dx, \tag{8}$$

where

$$f_{Laudau}(x|\mu,\sigma) = \frac{1}{\pi\sigma}\int_0^{\infty} e^{-t\frac{x-\mu}{\sigma} - \frac{2}{\pi}tlog(t)}\, dt. \tag{9}$$

More details on the critical value computation can be found in other works [1, 63].

Lemma 4.2 ([63]). *The probability of an FP detection by comparing p_{agg} with the critical value ξ in the HMP method is upper bounded by the detection threshold ϵ when all individual p-values are valid. A p-value is valid if it satisfies Lemma 4.1 from the ICAD framework.*

5 METHODOLOGY

An overview of the proposed OOD detection approach is as follows. As shown in Figure 3, at the training stage, the clustering method is used to filter the training set for prototypical datapoints. We call these prototypes *memories*. At runtime, we calculate the distance of an input with its closest memory and use it as the non-conformant score in the ICAD framework for computing the p-value of the input. The intuition for using this score is that an anomalous input is highly likely to lie far from the training set. For robust detection, we use the sliding window approach on a sequence

of inputs, where verdicts (or p-values) on individual datapoints in the sequence are combined by using the HMP method for OOD detection with a bounded false alarm rate.

5.1 Initializing the Memory Set

The intuition here is that high-dimensional data like images and LiDAR scans, which are generated from a real-world setting, cluster well in practice. The first step is to identify these broad categories in a quick and efficient fashion. One of the questions, however, is that the number of partitions to be made is often not known *a priori*. But drawing on the intuitions from an image distance metric, only small enough distances have perceptual meaning. Thus, the idea here is to populate the input space densely enough with memories such that every training point is within a threshold distance d of some memory. Algorithm 1 summarizes our approach. We pick a data point at random and then compute the distance score across all samples in the currently *RejectedSet* in a single linear pass. The data points which are *similar* enough are admitted as being close to a *memory*, and they are not considered as candidates for new memories in the next iteration. We continue this process until all data points are admitted into the set of memories M. This allows the subsequent algorithms to have a warm start. Algorithm 1 always terminates. This is because the *RejectedSet* decreases by at least 1 at each step. In the worst case, we have as many memories as the number of data points. But in most practical datasets, this is not the case.

ALGORITHM 1: Generate Initial Memories

Input: Dataset $\mathcal{S} : \{s_1, s_2, \ldots s_q\}$
Output: Memories $\mathcal{M} : \{m_1, m_2, m_3, \ldots, m_r\}$
Parameter: Distance Threshold d

$\mathcal{M} = \phi$
RejectedSet = $\mathcal{S}$
while RejectedSet $\neq \phi$ **do**
    s_m = pickRandomPoint(RejectedSet)
    for $s_i \in$ RejectedSet **do**
        if $\mathcal{D}(s_m, s_i) < d$ **then**
            RejectedSet = RejectedSet $\setminus s_i$
    $\mathcal{M} = \mathcal{M} \cup \{s_m\}$
return $\mathcal{M}$

5.2 Learning Memories

To restate, we are given a dataset $\mathcal{S}$, with q elements, and we wish to compute an r size memory set $\mathcal{M} = \{m_1, m_2, \ldots, m_r\}$ with certain desirable properties. The search for memories can be simplified by viewing this as a search through a graph $\mathcal{G}$ [43] with subsets $\mathcal{S}_r \subset \mathcal{S}$ as its nodes. Each subset of size r defines a choice for the memory set $\mathcal{M}$.

Definition 5.1 (Memory Search Graph $\mathcal{G}$). The undirected graph $\mathcal{G}$ is represented by an ordered pair (V, E). The set of nodes V is the collection of subsets of original dataset $\mathcal{S}_r \subset \mathcal{S}$. An edge $e \in E$ exists between two nodes $\mathcal{S}_r^1$ and $\mathcal{S}_r^2$ iff $|\mathcal{S}_r^1 \cap \mathcal{S}_r^2| = r - 1$. In other words, they differ by at most one memory.

Each node of the graph has an associated cost given by Equation (1). Hence, starting from some node, it is possible to visit neighboring nodes with decreasing costs in the search process. What we present next is a combination of *Global* resets and *Local* minimization to approximate the optimal choice.

ALGORITHM 2: Generate Memories

Input: $\mathcal{S} : \{s_1, s_2, \ldots s_q\}$
Output: Memories $\mathcal{M} : \{m_1, m_2, m_3, \ldots, m_r\}$
Parameter: (Max Global Steps : Z_g, Max Local Steps : Z_l, Distance Threshold d)

```
1:  BestCost = ∞
2:  for 1 ≤ g ≤ Z_g do
3:      Memory Set M = GenerateInitialMemories(S, d)      ▷ Pick an initial set of memories
4:      G = CreateGraph(S, |M|)                           ▷ The memory search graph
5:      v = FindNode(M, G)
6:      CurrentCost = ComputeCost(v)
7:      for 1 ≤ l ≤ Z_l do                                ▷ Compute incremental improvements
8:          v' = PickNeighbor(v, G)
9:          NewCost = ComputeCost(v')
10:         if NewCost < CurrentCost then
11:             v ← v'
12:             CurrentCost ← NewCost
13:     if CurrentCost < BestCost then
14:         BestCost = CurrentCost
15:         M = M                                         ▷ Store the best memory set observed
16: return M
```

Algorithm 2 picks the eventual memories used in OOD detection. Similar to the standard CLARANS algorithm, each node in $\mathcal{G}$ has $r(q-r)$ neighbors, where r is the number of memories. The number of neighbors can be quite large given the scale of modern machine learning datasets with large q. What we do here is start with a reasonable choice for initial node in $\mathcal{G}$ and then greedily look for local improvements for a fixed number of iterations. The global search starts by using Algorithm 1, to generate the initial set of memories as node v in $\mathcal{G}$. Notice that we do not choose the number of memories *a priori* but instead gets picked as a consequence of *distance score d*. The partitioning cost for the choice of memories is computed by the function *ComputeCost* which evaluates Equation (1). Note that this can be expensive since it needs a total of $r \times q$ distance computation operations. The *local* search (lines 7–12) implements a greedy strategy to pick the neighborhood node that produces a descent. The outer loop of the algorithm keeps track of the node with the minimum cost for each such reset produced in line 3. Algorithm 2 trivially terminates, as each search proceeds for a fixed number of steps.

Definition 5.2 (Memory System $\mathcal{M}_{\mathcal{S}}$). A memory system is a collection of pairs $\mathcal{M}_{\mathcal{S}} := \{(m_1, q_1), (m_2, q_2) \ldots, (m_r, q_r)\}$, where

$$\begin{aligned} q_i &= |Q_{m_i}| \\ Q_{m_i} &= \bigcup_{s \in \mathcal{S}} \mathbb{I}(m_i = \underset{i}{argmin}\ \mathcal{D}(s, m_i)). \end{aligned} \tag{10}$$

The memory-based OOD detector does not need q for OOD detection, so we simplify the memory system as $\mathcal{M}_{\mathcal{S}} := \{m_1, m_2 \ldots, m_r\}$.

5.3 Scaling Memory Search

Even though the number of memories produced in Algorithm 2 might be small enough compared to the full dataset $\mathcal{S}$, a search through the list of memories might still be challenging. To remedy this potential drawback, we deploy a simple hashing technique first introduced in the work of

Fukunaga and Narendra [22]. The distance metric $\mathcal{D}$ discussed in Section 4.2 was a proper distance metric, which implies that the distance function respects triangle inequality. In what follows, we describe a possible avenue to speed up the search for the k nearest memories, the intuition being that for sufficiently different memories, computing a single distance pair can be used to reject other memories from further consideration.

We are interested in computing the nearest neighbor—that is, $k = 1$ in the set $\mathcal{M}_{\mathcal{S}}$ for a test point x_t. Assume that we wish to compute the distance between a test point x_t, and some memory m_j and the distance $\mathcal{D}(x_t, m_i)$ is known. Then in the triangle formed by the triplet (m_i, x_t, m_j), the following two equations are true:

$$\mathcal{D}(m_i, x_t) - \mathcal{D}(m_i, m_j) \leq \mathcal{D}(m_j, x_t)$$

and

$$\mathcal{D}(m_i, m_j) - \mathcal{D}(m_i, x_t) \leq \mathcal{D}(m_j, x_t),$$

meaning that $\mathcal{D}(m_j, x_t)$ is lower bounded by $|\mathcal{D}(m_i, m_j) - \mathcal{D}(m_i, x_t)|$. If we are interested in memories which are within a certain threshold (say h) of m_j, we do not actually need to compute the distance $\mathcal{D}(m_j, x_t)$ if the following equation holds *True*:

$$|\mathcal{D}(m_i, m_j) - \mathcal{D}(m_i, x_t)| > h. \tag{11}$$

For each memory m_i, we can pre-compute a look-up table for the inter-memory distance Q: $\{(m_j, \mathcal{D}(m_i, m_j)) | 1 \leq j \leq |\mathcal{M}_{\mathcal{S}}|, j \neq i\}$. This can lead to reduction in the search space in practice by pruning out memories from further consideration each time the distance of a memory from x_t gets measured. For $k > 1$, similar reasoning holds. The only difference being that in this case, the search algorithm tracks the distance of the k^{th}-memory furthest from the test point.

5.4 Detecting Distribution Shifts

To summarize, we know how to go from the set of training data $\mathcal{S}$ to the set of memories $\mathcal{M}_{\mathcal{S}}$. It is generated by a smart initialization of the set of memories (Algorithm 1), followed by a refinement using the medoid-based partitioning technique discussed in Algorithm 2. Additionally, to handle any potential slowdowns, we briefly discussed how one can use the inter-memory distance to prune out large parts of the search space. This allows the system to scale to larger memory systems. Now, we discuss the proposed algorithm for detecting distribution shifts in real time.

In practical scenarios, detecting a distribution shift requires a robust mechanism. We achieve this by using a sliding window based approach to track the number of OOD datapoints. Algorithms 3 and 4 summarize the offline and real-time stages of the proposed OOD detection algorithm, respectively. The real-time detection Algorithm 4 is based on the HMP method for combining p-values (computed from the ICAD framework) of individual datapoints in a time-series window. Given the desired FP rate ϵ and the size of sliding window n, the critical value ξ is computed in line 1 of the algorithm. The SSIM distance between the input and the closest memory from the memory system M is used as the non-conformity score for calculating the p-value of input x_i in line 4.[2] The aggregated p-value p_{agg} on the sliding window of the input datapoints $(x_{i-n}, x_{i-n+1}, \ldots x_i)$ is computed by using the HMP method in line 6 of Algorithm 4. If p_{agg} is less than the ξ, then the window is labeled as OOD.

Theorem 2. *The probability of misdetecting an iD window W as OOD by Algorithm 4 is upper bounded by the desired FP rate ϵ.*

[2]Non-conformity scores of the calibration datapoints are also computed in the offline stage (Algorithm 3) by using the same SSIM distance of the calibration datapoints and the closest memory in M.

ALGORITHM 3: Offline Stage

Input: A set of iD time-series traces $\mathcal{T} : \{T_1, T_2, \dots\}$
Output: Calibration Set Scores $C : \{\alpha_1, \alpha_2, \dots, \alpha_m\}$, Memory Systems $\mathcal{M}$

1: Shuffle and split $\mathcal{T}$ into a proper training trace and calibration traces
2: Prepare a calibration set $(c_1, c_2, c_3, \dots, c_m)$ by randomly sampling m datapoints from the calibration traces
3: Generate the set $\mathcal{M}$ of memories by using Algorithm 2 on the set $\mathcal{S}$ of all datapoints in the proper training traces
4: **for** each $c_i \in (c_1, c_2, c_3, \dots, c_m)$ **do**
5: compute a non-conformity score of c_i: $\alpha_i \leftarrow SSIM_{dist}(c_i, \mathcal{M})$
6: $C \leftarrow C \bigcup \{\alpha_i\}$
7: **return** $\mathcal{M}, C$

ALGORITHM 4: Real-Time Detection of an OOD Window in Time-Series Data

Input: Input test sequence $(x_1, x_2, \dots)$, Memory Systems M, Calibration Set Scores $C : \{\alpha_1, \alpha_2, \dots, \alpha_m\}$, the desired FP rate $\epsilon \in (0, 1)$, window length n
Output: 1 on detection of an OOD window

1: Compute critical value $\xi \leftarrow (\epsilon, n)$ from Equation (8)
2: **for** $x_i \in$ input sequence **do**
3: compute non-conformity score of x_i: $\alpha_i \leftarrow SSIM_{dist}(x_i, M)$
4: $p_i \leftarrow \frac{|\{j=1\dots m : \alpha_i \le \alpha_j\}|+1}{m+1}$
5: **if** $i \ge n$ **then**
6: Compute $p_{agg} \leftarrow HMP(p_{i-n}, p_{i-n+1}, \dots p_i)$
7: **if** $p_{agg} < \xi$ **then**
8: **return** 1
9: **return** 0

Proof. If an individual datapoint x in W is drawn from the training distribution D, then x and datapoints in the calibration set are independent and identically distributed with respect to D. The p-value of x computed in line 4 of the algorithm is uniformly distributed and therefore valid according to Lemma 4.1. For all iD datapoints $(x_{i-n}, x_{i-n+1}, \dots x_i)$ in W, p_{agg} is computed from their valid p-values in line 6. ξ is computed from the desired FP rate ϵ in line 1. The probability of misdetecting W as OOD by comparing with it ξ is therefore bounded by ϵ according to Lemma 4.2. □

5.5 Heatmap Generation Algorithm

The SSIM metric and the memory system can be used to generate explanations as promised. To elucidate this aspect, first we point the reader to Equation (5), for SSIM. It is essentially an aggregate of the local features around each individual pixel. This means that when dissimilarity arises, it is possible that a few pixels account for the major differences. Highlighting such pixels can produce a reasonable feedback to the system designer. For a test image that is sufficiently different, it is still true that there is some memory it finds itself being closest to. We use this closest memory to generate an explanation. The details are presented in Algorithm 5. It takes in as input a sample test image and its closest memory, and produces a feedback image. The first step is to create an SSIM map D_x on line 2 using *ComputeFullSSIM*. Next, it iterates through the pixel locations of the SSIM map and alters the color of pixels with the high contributions to the SSIM matrix. This

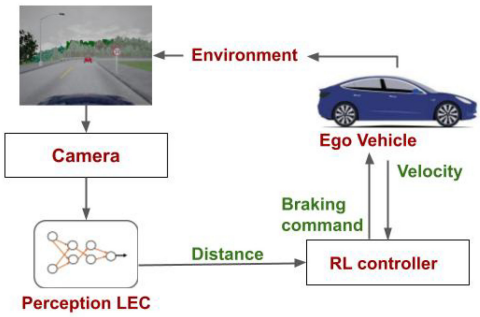


Fig. 4. Closed loop of the AEBS from Cai and Koutsoukos [11].

is achieved through simple thresholding in line 5. The array x'_t is the visualization of the most dissimilar parts between the test frame and its closest memory. Note that the best explanation that explains a certain behavior is often context specific. Here we find highlighting the important pixels to be the more useful one.

ALGORITHM 5: Heatmap Generation

Input: Time-Series Data Point $x_t \in \mathbb{R}^{a \times b}$, Closest Memory m_c
Output: Heatmap $x'_t \in \mathbb{R}^{a \times b}$
Parameter: Color Distance Threshold d_{color}

1: Instantiate $x'_t \leftarrow x_t$
2: $D_x \in \mathbb{R}^{a \times b} \leftarrow$ ComputeFullSSIM(x_t, m_c)
3: **for** $1 \leq a' \leq a$ **do**
4: **for** $1 \leq b' \leq b$ **do**
5: **if** $D_x[a', b'] > d_{color}$ **then**
6: $x'_t[a', b'] \leftarrow$ PaintPixel$(x_t[a', b'])$
7: **return** x'_t

6 CASE STUDY 1: SIMULATED SCENARIO FOR AUTONOMOUS DRIVING SYSTEMS

6.1 System Description

In this section, we consider a driving scenario with an AEBS as described in the work of Cai and Koutsoukos [11]. The AEBS is designed for preventing the potential collisions by detecting the front obstacles or monitoring the behavior of a leading car. This closed loop of AEBS (Figure 4) consists of a perception-based LEC and a controller trained with the **Reinforcement Learning (RL)** algorithms. Given an input image from the front camera, LEC is used for estimating distance between the ego vehicle and its front obstacles. The estimated distance is used as an input to the RL-based controller for determining the braking control action by the ego vehicle. Driving simulators such as CARLA [15] are a convenient source for effectively training LECs and RL-based controllers. Thus, in this case study, we evaluate our OOD detection approaches on simulated scenarios.

iD Data. A simulated driving dataset in the work of Cai and Koutsoukos [11] is generated by using an open source autonomous driving simulator, CARLA [15]. The iD traces are simulated in daytime driving scenarios with light rain (or low precipitation level) where the front obstacle is always a car. The dataset contains 33 training traces that include 4,488 images with a precipitation

Table 1. OOD Detection Results on Heavy Rain Traces

	(n, ϵ, d)	Mem	AUROC	FP	FN	Avg Delay	Exec Time (ms)
Ours	(6,0.01,0.2)	141	**97.22**	**0/26**	**0/74**	**0.0**	**19.18**
VAE [11]	N/A	N/A	89.08	2/26	**0/74**	**0.0**	55.95
β-VAE [54]	N/A	N/A	48.00	**0/26**	74/74	N/A	106.8

level from 0 to 10 (inclusive). The average length of the training traces is about 120, and the sampling rate is 20 Hz. More details about the system setup and training can be found in the work of Cai and Koutsoukos [11].

Types of OODs. There are six different types of distribution shifts: distribution shifts by increasing the precipitation level (leading to heavy rain), distribution shifts by fog, snow and brightness change, distribution shifts by unseen front obstacles, and distribution shifts by adversarial perturbations.

Evaluation Metrics. In our experiments, we define iD traces as negative and OOD traces as positive. Thus, the FP rate is the number of iD traces that were detected as OOD traces, and the **False-Negative (FN)** rate is defined as the number of OOD traces falsely detected as iD traces. Additionally, we report the average time delay in OOD detection. In other words, if a trace becomes OOD at a certain time t, the delay in detection is computed as the difference between the time of detection and t. The average delay is the average detection delay on all successfully detected OOD traces.

Comparison with the Baselines. We compare our results with the VAE-based state-of-the-art detectors by Cai and Koutsoukos [11] and Sundar et al. [54]. We will be using the shorthand 'VAE' for the detector of Cai and Koutsoukos [11] and 'β-VAE' for the detector of Sundar et al. [54]. These baselines report their results on traces by using a point-based detection approach where an OOD detection alarm on the trace is raised on detection of the first OOD datapoint. For a fair comparison with these baselines, we also report our results on traces in a similar manner. We raise an OOD detection alarm on the trace on detection of the first OOD window in the trace. All detectors are trained on the same training traces and calibration traces. Both our detector and baselines are trained on an NVIDIA Quadro RTX 6000.

6.2 OOD Detection for Distribution Shift Due to Change in Weather and Lighting

The weather and lighting changes in driving scenarios have proved to be challenging for autonomous vehicles with image inputs [55]. In this section, we evaluate our approach on distribution shifts introduced by heavy rain, fog, snow, and darkness. Note that for all OOD traces, the level of weather/lighting factors (e.g., precipitation level, darkness) gradually increase with time. In our experiments, we do a 40/60 split on the proper training traces and calibration traces, respectively, and the size of the calibration set is 1,000 datapoints.

6.2.1 Distribution Shift Due to Heavy Rain. Following the experiments in the work of Cai and Koutsoukos [11], we aim for OOD detection due to an increase in the iD precipitation level. The traces with heavy rain, specifically with precipitation levels greater than 20, are called *OOD traces*. There are 100 test traces in total, of which 26 of them are iD traces and the rest are OOD traces. Note that our algorithm contains a few hyperparameters including distance for learning a memory system d, window length n, and desired FP rate on window ϵ. These hyperparameters can be set by the users to achieve a certain level of safety according to the application of interest.

As shown in Table 1, we are able to detect all heavy rain traces with a small detection delay. None of the iD traces are detected as OOD by our approach. The performance of our detection algorithm

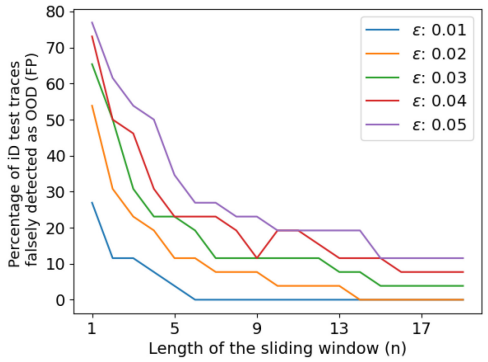


Fig. 5. Percentage of iD test traces falsely detected as OOD (FP) versus the length of the sliding window n. Each line shows results with a different expected FDR $\epsilon \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$. Note that for all settings in these plots (i.e., for all values n and ϵ), our detector is able to detect all OOD traces, resulting in 0 FN detection.

is better than the performance of both VAE-based detectors. We also observe that both of these baselines are required to select their detection threshold from the validation data for controlling the FP rate, whereas the memory-based OOD detector uses the user-specified ϵ as the desired upper bound on the FP rate. The training time of VAE and β-VAE is approximately 3 hours, and the time of learning a memory set for the same training dataset is about 10 minutes. Another noticeable result is that Algorithm 2 efficiently compresses approximately 2,000 proper training images into 141 memories. We also report the average execution time by our algorithm on test frames and show that our algorithm runs in real time since it is well below sampling time of the system in Table 1. Our detector also has better running speed compared to the baselines.

We also perform an ablation study of our detector on the window length n used for detection. Figure 5 compares the FP detection rate on different window lengths. We observe that the FP rate decreases with the increase in the window length, which indicates that larger windows better control the FP rate. This justifies the use of sliding windows for robust detection instead of using only a single datapoint (or window length equal to 1) for detection. Additionally, for a given window length, the FP rate decreases as the ϵ decreases, which is as expected.

6.2.2 Distribution Shift Due to Fog, Snow, and Low Lighting Conditions. Here, we want to detect the distribution shifts due to fog or change in lighting conditions from the iD driving scenarios. We generate 27 foggy/snowy traces, where the level of fog/snow gradually increases with time. Similarly, we generate 27 night traces to mimic the scenario where the day starts getting darker. We show an example of a foggy trace in Figure 6(b) and an example of a night trace in Figure 6(c). To perform OOD detection for these three scenarios, we use the same hyperparameters that we use for detection due to heavy rain in Section 6.2.1. As shown in Table 2, our detector is able to detect all OOD traces due to both foggy and night-time scenarios with a small detection delay. Our algorithm is also able to detect all snowy traces but with a longer delay compared to the VAE method because that our detector does not raise an alarm before viewing a complete sliding window to avoid over-sensitivity to one frame.

6.3 OOD Detection for Distribution Shift Due to Change in the Front Obstacle

For AEBS, one challenge is that the LEC works only for distance estimation from the front obstacles seen during its training. Considering that the LEC is only trained with a car as the front obstacle,

(a) An example of a light rain (in-distribution) trace

(b) An example of a light rain combined with foggy weather

(c) An example of a light rain combined with night setting

Fig. 6. Example of iD traces (light rain) and OOD traces (foggy and night).

Table 2. OOD Detection Results on Night, Foggy, and Snowy Traces

	Night Traces			**Foggy Traces**			**Snowy Traces**		
	AUROC	**FN**	**Avg Delay**	**AUROC**	**FN**	**Avg Delay**	**AUROC**	**FN**	**Avg Delay**
Ours	**97.50**	0/27	**0.0**	**99.70**	**0/27**	**0.07**	**100.0**	**0/27**	6.0
VAE	96.42	0/27	0.11	95.54	**0/27**	6.41	99.02	**0/27**	**1.29**
β-VAE	**97.50**	0/27	0.26	29.09	27/27	N/A	99.17	27/27	N/A

Note that the hyperparameters $\boldsymbol{n}$, $\boldsymbol{\epsilon}$, $\boldsymbol{d}$ are the same as in heavy rain OOD detection.

it can always provide an accurate distance estimation and avoid collisions at a safe distance from the leading car. However, if the front obstacle changed to a bike, the LEC will fail to make a correct distance estimation, leading to a crash with the bike. In this experiment, the unseen object in the front is considered as the distribution shift, and we show an example of such a distribution shift in Figure 7 where the ego vehicle crashes into the bike. We generate 27 OOD traces with a bike as the front obstacle in the same environmental conditions as the training. Again, we use the same set of hyperparameters as in heavy rain OOD detection and show results in Table 3.

6.4 Bounded FDR

We report box plots of the FDR with respect to the detection threshold ϵ on the iD test windows of clear weather. These plots are shown in Figure 8. We randomly split the calibration and test traces and get the FDR with different values of ϵ from 0.01 to 0.2 with the step size of 0.01. This is repeated 10 times, and we show box plots with the mean and inter-quartile range. The plot shows that the false detection by the proposed detector on the (sliding) iD test windows is bounded by the FDR ϵ on average.

(a) Ego vehicle stopping at a safe distance from the lead car at test time

(b) Shift from training distribution with a biker as the front obstacle leads to a crash at test time

Fig. 7. Illustration of a safety hazard (i.e., collision due to shift from the training distribution).

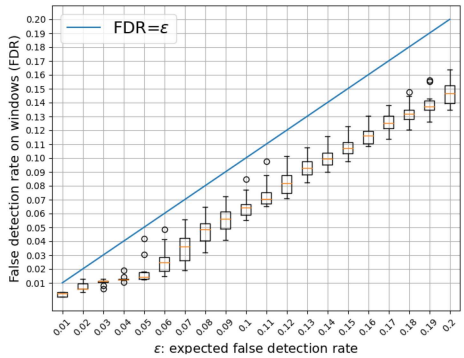


Fig. 8. FDR is bounded by the expected FDR ϵ on average ($n = 6, d = 0.2$).

Table 3. Detection Results on OOD Traces with Bikes as the Front Obstacles

	FP	FN	Avg Delay
Ours	**0/26**	**0/27**	**0.0**
VAE	2/26	**0/27**	0.85
β-VAE	**0/26**	27/27	N/A

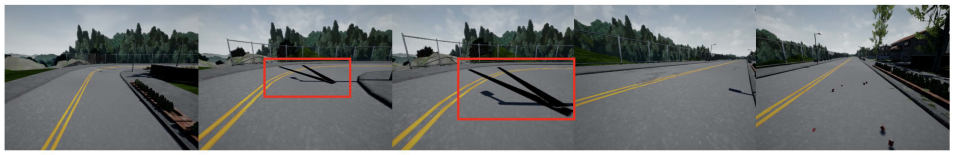
Note that the hyperparameters $\boldsymbol{n}$, $\boldsymbol{\epsilon}$, $\boldsymbol{d}$ are the same as the OOD detection due to heavy rain.

6.5 OOD Detection for Distribution Change Due to Adversarial Perturbations

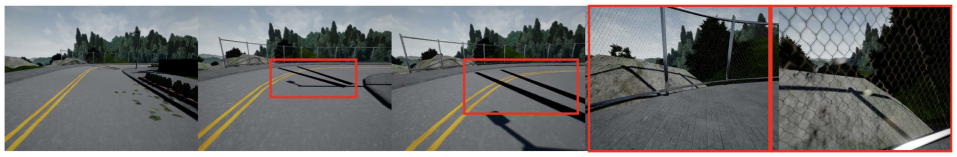
Previous work [9] proposed a physical adversarial attack by painting lines on the road to confuse the autonomous driving system (highlighted in the red box of Figure 9(b)). As shown in Figure 9(c), this attack misleads the car to follow the painted lines, leading the car to crash into the fence [11]. We use the self-driving dataset with painted lines generated by Cai and Koutsoukos [11], where evaluation is focused on the right corner driving cases. A clean (or non-attacked) trace is shown in Figure 9(a).

All OOD traces contain OOD frames due to painted lines, but in some of these OOD traces, the car successfully takes the right turn without crashing into the fence. One such example of this OOD trace without the crash is shown in Figure 9(b). For OOD traces with a crash, instead of taking

(a) A train trace (clean)



(b) A test trace (adversarial sticker on the road but does not end with a crash)



(c) A test trace (adversarial sticker on the road but ends with a crash against a fence as shown in the last two images)

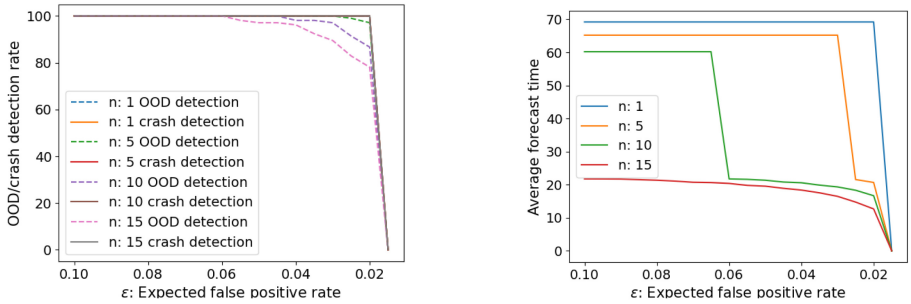
Fig. 9. OOD-ness due to adversarial road perturbations [11].

the right turn, the car crashes into the fence. One such example of this OOD trace with a crash is shown in Figure 9(c). Unlike previous experiments, there is no label of which frame in a trace is an OOD frame, so we report the successful crash prediction rate. A successful crash prediction on a trace is defined as predicting an OOD window before the crash actually happens. In addition to the OOD detection rate and crash detection rate, we also want to evaluate the forecast time of crashes by using our OOD detection technique. Assuming that the crash happens at time t and the detection time is t_p, we define forecast time to be $t - t_p$. We report the average forecast time for all successfully predicted crash traces. There are 105 OOD traces in total, and 64 of them result in a crash. Table 4 and Figure 10 show the performance of our detector on the distribution shift due to adversarial attack of the painted lines.

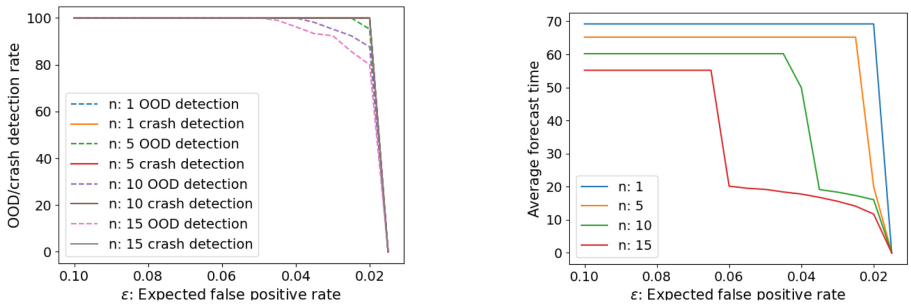
Table 4 shows that we are able to detect all OOD traces and predict crashes before they actually occur around 60 timesteps prior. We observe that although both VAE and β-VAE successfully detect all OOD traces and avoid the crash beforehand, our method has a better average forecast time compared to VAE and a shorter execution time compared to both baselines. Figure 10 shows the performance of OOD/crash detection rate with respect to the expected FP rate (ϵ) for different window lengths (n). We observe that as ϵ decreases, the OOD/crash detection rate decreases and the average forecast time also reduces (the higher the better). This is expected because the memory-based detector becomes less conservative, where the detectors will try to detect most of the traces as iD. The results on OOD detection deteriorate with decrease in ϵ. Additionally, we observe that with decrease in ϵ, although the OOD detection rate drops slightly, our predictor is still able to predict all crashes beforehand. This is reasonable because OOD traces with a crash deviate more from the iD traces as compared to the OOD traces without a crash as illustrated in Figure 9(b) and (c). This shows the robustness of our sliding -window based approach compared to single-point-based methods, which avoids being sensitive to minor deviations from iD data. Figure 11 shows

Table 4. OOD Detection Results on Adversarial Sticker Detection

	(n, ϵ, d)	Mem	OOD Detection Rate	Crash Detection Rate	Avg Forecast (frame)	Exec Time (ms)
Ours	(10,0.07,0.5)	69	100.0	100.0	60.19	24.74
Ours	(10,0.05,0.6)	26	100.0	100.0	60.19	**19.21**
VAE	N/A	N/A	100.0	100.0	21.41	44.95
β-VAE	N/A	N/A	100.0	100.0	**69.19**	103.44



(a) Detection performance vs detection threshold (Memory distance (d): 0.5)



(b) Detection performance vs detection threshold (Memory distance (d): 0.6)

Fig. 10. OOD traces detection results for detecting adversarial attack/crashes on the road with hyperparameter sweeping.

the performance of OOD/crash detection rate with respect to the detection threshold for VAE and β-VAE. Here we observe that the crash detection rate generally drops quicker with the decrease in the detection threshold, especially for β-VAE, making it harder to pick an appropriate detection threshold for detecting a crash.

6.6 OOD Reasoning Using the Closest Memory

Although the previously proposed VAE-based OOD detectors [11, 19] achieve comparable performance for the simulated autonomous driving scenarios, such detectors do not provide any explanation for the distribution shift with its detection. As described in Section 5.5, in addition to the detection, we provide an interpretation of our OOD detection result. When the detector reports a test frame as an OOD frame, we know that there is no matching memory from the training data. In addition to this quick intuition, our detector can further provide reasoning at the pixel level.

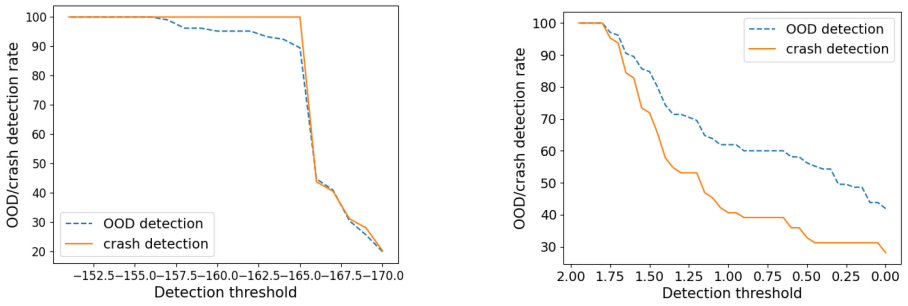


Fig. 11. OOD traces detection results for detecting adversarial attack/crashes using VAE-based OOD detectors. Detection performance vs detection threshold for VAE (left) and β-VAE (right).

(a) Match the input test image with memories in training data

(b) Highlight the least similar part compared to the memory

Fig. 12. OOD detection reasoning for adversarial sticker detection.

We achieve this by selecting the memory with the smallest distance and generate a heatmap to highlight the OOD part for the test input. If the test input is far from all of the memories, the resulting highlighted pixels can be used to alarm drivers. The interpretability of our approach could assist drivers in deciding whether to switch or continue driving on the autonomous mode instead of solely relying on the detection results.

We show the highlighted OOD pixels after detection in Figure 1(b) and Figure 12(b). As shown in Figure 1(b), the unrecognized biker is highlighted in this OOD frame. In Figure 12 for the detected adversarial attacked frame, we highlight the pixels that contain the adversarial stickers on the road.

7 CASE STUDY 2: DRIVING WITH LIDAR

7.1 System Description

LiDAR is an important component for building safe autonomous systems due to its ability to construct a comprehensive three-dimensional model of the surroundings. It has the ability to reliably complement a camera module when the system needs faster and more detailed feedback on the environment. LiDAR sensors on an autonomous car compute the relative position of the closest obstacle at a certain angle. This is achieved by measuring the time a reflected light beam takes to return to the LiDAR sensor after being emitted from the source. Ivanov et al. [25] discuss a challenging setup for steering a car with LiDAR inputs and a neural network controller. Although the

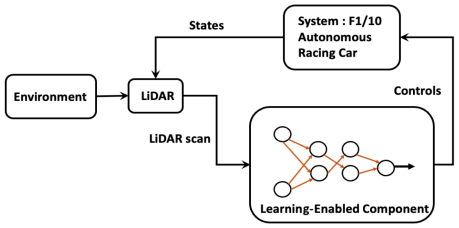


Fig. 13. System description of the F1/10 autonomous car for navigation.

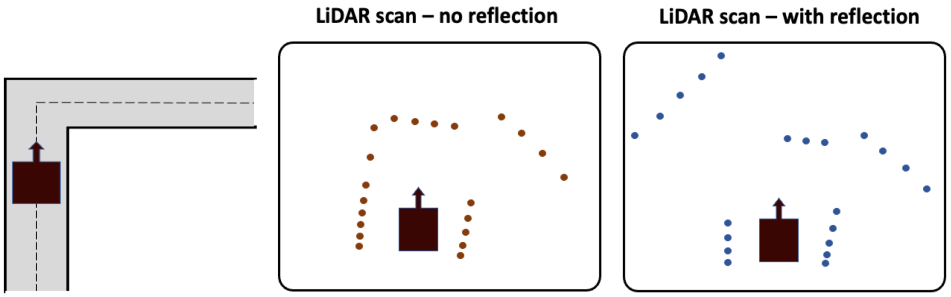


Fig. 14. Left: We show a setting where the car should take a right turn on an L-shaped track. Middle: The dots show the distance estimates as provided by the sensor. It matches well with the position of the obstacles. Right: Due to reflection from the left wall, it gives a false impression of no obstacle to the left of the car when deployed in the real world.

inputs are not as complicated as image inputs from camera, the LECs trained with LiDAR inputs still suffer a failure in an OOD setting.

We show the setup of the F1/10 autonomous car from Ivanov et al. [25] in Figure 13. This closed-loop system consists of an F1/10 autonomous driving car from F1Tenth [3]. The LEC in this case is a neural network controller that decides the control action for the following step. The LiDAR measurements are the inputs to the F1/10 autonomous car, and the system states include velocities and positions of the vehicle. The neural network controller is trained with standard deep RL techniques like DDPG (deep-deterministic policy gradient) and TD3 (Twin Delay DDPG) [40]. The hidden layers are of size 64×64 and 128×128, respectively. The controller is designed for navigating in a structured environment like an L-shaped track shown in Figure 14. The LiDAR sensors have 1,081 rays at maximum and range of 5 m. The LiDAR scans on the system sweep from -135 degrees to 135 degrees where 0 degree indicates the front heading of the vehicle.

7.2 Simulation vs Reality

Simulation serves as an important resource for training LECs due to the cost and effort of obtaining a large real-world dataset. However, when deploying the system to the real world, the simulation-to-real gap is always a challenge and sometimes causes an unpredicted behavior for safety-critical applications. Figure 14 illustrates such a failure where the car intends to turn right at the left corner of an L-shaped track from Ivanov et al. [25]. The reflective surface of the environment causes a delayed response of rays, and missing measurements make the system believe that there is no obstacle at a certain angle. The neural network controllers are shown to be sensitive to those inputs and fail to make the right steering commands. Although verification techniques in other works [16, 26] could provide safety properties of the systems, the assumption is that the

Table 5. OOD Detection for LiDAR Data

	(n, ϵ, d)	Mem	TPR	FPR	MPR	Avg Forecast	Exec Time (ms)
Ours	(10,0.001,0.2)	41	85.71	35.53	12.5	22.73	3.7
Ours	(10,0.001,0.3)	24	91.07	40.7	8.93	24.5	**1.99**
Ours	(20,5e-4, 0.3)	24	75.0	**17.54**	14.07	13.74	**1.99**
VAE	N/A	N/A	85.71	77.23	8.93	120	11.27
β-VAE	N/A	N/A	**98.21**	76.39	**1.79**	**130**	48.01

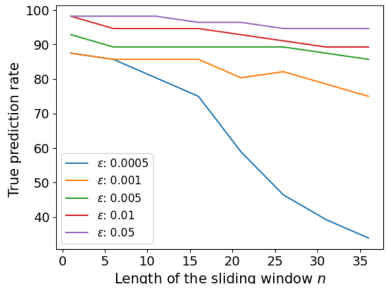
car operates in a certain expected environment. If the car could receive an alarm of OOD LiDAR measurements before the crash happens, then it could take actions or change mode to avoid being misguided by a neural network controller.

7.3 Avoiding a Crash with OOD Detection

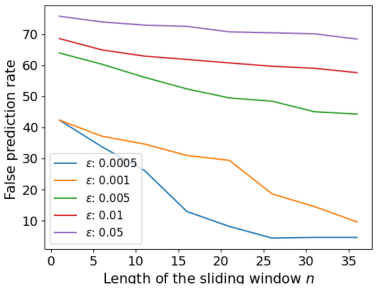
In this section, we report our experimental results on predicting crash when driving with LiDAR measurements. Ivanov et al. [25] observed that the delayed response of LiDAR rays due to reflection are correlated with crashes. We hypothesize that the distribution shifts of the LiDAR input could be due to a deviation from simulation data (ideal and no reflections) to real-world data. Hence, we define the distribution shift in this setup as the LiDAR scan with reflected rays. However, there is no label on the LiDAR measurements which contain rays obtained from reflective surfaces. Therefore, we evaluate our approach by predicting a crash beforehand. This achieved by flagging inputs which deviate from iD data. This is also an interesting case study to evaluate whether the OOD detection could be used for predicting a crash. The LiDAR dataset $\mathcal{S}$ is defined as $\{T_1, T_2, T_3 \dots\}$ and $T_i = \{(x_1, p_2), (x_2, p_2), \dots\}$, where $x_i \in \mathbb{R}^q$ is the LiDAR scan and p_i indicates whether there is a crash at timestep i. As described in the work of Ivanov et al. [25], the training data is obtained from a simulator for the 12 different controllers, and no crash occurs in simulation. The SSIM distance metric is still applicable in this case study. We created a two-dimensional input for LiDAR measurements by repeating the one-dimensional measurements. There are 8 training traces, 10 calibration traces, and 236 test traces in our experiments (55 of them show a crash). The average length of the test traces is about 410. The size of our calibration set is 4,000 scans. A crash prediction is successful if an OOD alarm is raised before the real crash happens. Our predictor reports a trace as an OOD trace if at least one sliding window in a given trace is flagged as an OOD according to Algorithm 4. The evaluation metrics are defined as follows:

$$\begin{aligned}
\textbf{True Prediction Rate (TPR)} &= \frac{\#\text{ crash predicted successfully}}{\#\text{ real crash happens}} \\
\textbf{False Prediction Rate (FPR)} &= \frac{\#\text{ no crash happens but predicted with crash}}{\#\text{ crash predicted}} \\
\textbf{Missed Prediction Rate (MPR)} &= \frac{\#\text{ crash happens without alarm}}{\#\text{ real crash happens}}
\end{aligned} \tag{12}$$

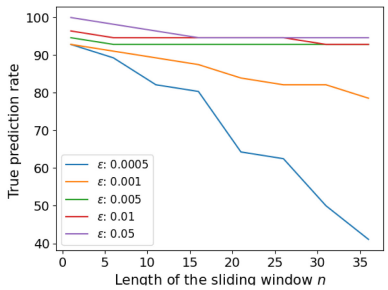
According to the results reported in Table 5 and Figure 15, our predictor is able to detect 75% of the crashes with $\approx$ 14 timesteps ahead in the best case. The FP rate is around 18%, and the MPR is about 14% for the best choice of hyperparameters. Compared to the two baselines, we have a comparable TPR that has a much lower false crash prediction rate. Our detector also has better execution time. As shown in Figure 15, with the same window size, increasing ϵ results in both the correct crash prediction rate and FPR getting higher. We also observe that the correct crash prediction rate and FPR decrease as the size of sliding window increases.



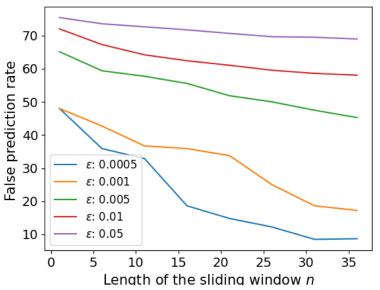
(a) True prediction rate vs sliding window size (Memory distance: 0.2)



(b) False prediction rate vs sliding window size (Memory distance: 0.2)



(c) True prediction rate vs sliding window size (Memory distance: 0.3)



(d) False prediction rate vs sliding window size (Memory distance: 0.3)

Fig. 15. OOD detection results for detecting LiDAR crash with different hyperparameters.

8 CONCLUSION

OOD detection can be of utmost importance in ensuring the safety of CPS equipped with LECs. In this article, we proposed a memory-based OOD detector to detect distribution shifts for a real-time system. Our algorithm is able to achieve state-of-the-art results in OOD detection for self-driving car applications with interpretability and statistical guarantees, without compromising execution times. In the future, we would like to extend this technique on applications beyond self-driving cars where anomalous inputs are challenging to handle.

ACKNOWLEDGMENTS

We would like to thank Professor Radoslav Ivanov at Rensselaer Polytechnic Institute for discussions regarding the LiDAR experiments and for sharing the data. We are grateful to Professor Edgar Dobriban at the University of Pennsylvania and Shuo Li at the University of Pennsylvania for their insightful discussions regarding the application of HMP techniques and multiple hypothesis testing. We thank the reviewers for their valuable and constructive feedback that helped to improve the article.

REFERENCES

[1] D. Ter Haar (Ed.). 1965. On the energy loss of fast particles by ionisation. In *Collected Papers of L. D. Landau*. Pergamon, 417–424. https://doi.org/10.1016/B978-0-08-010586-4.50061-4

[2] Leonard Kaufman and Peter J. Rousseeuw. 1990. *Partitioning Around Medoids (Program PAM)*. Wiley Series in Probability and Statistics. Wiley, 68–125. https://doi.org/10.1002/9780470316801.ch2

[3] F1TENTH. 2021. F1TENTH Home Page. Retrieved February 15, 2024 from https://f1tenth.org

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved February 15, 2024 from https://www.tensorflow.org/

[5] Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. 2017. An information-theoretic analysis of deep latent-variable models. *CoRR abs/1711.00464* (2017). http://arxiv.org/abs/1711.00464

[6] Vineeth Balasubramanian, Shen-Shyang Ho, and Vladimir Vovk. 2014. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*. Newnes.

[7] Michele Basseville and Igor V. Nikiforov. 1993. *Detection of Abrupt Changes: Theory and Application*. Vol. 104. Prentice Hall, Englewood Cliffs, NJ.

[8] Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. 2021. Testing for outliers with conformal *p*-values. *arXiv preprint arXiv:2104.08279* (2021).

[9] Adith Boloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. 2020. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture* 110 (2020), 101766. https://doi.org/10.1016/j.sysarc.2020.101766

[10] Dominique Brunet, Edward R. Vrscay, and Zhou Wang. 2012. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing* 21, 4 (2012), 1488–1499. https://doi.org/10.1109/TIP.2011.2173206

[11] Feiyang Cai and Xenofon Koutsoukos. 2020. Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In *Proceedings of the 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS '20)*. IEEE, 174–183.

[12] Derek Caveney. 2010. Cooperative vehicular safety applications.. *IEEE Control Systems Magazine* 30, 4 (2010), 38–53. https://doi.org/10.1109/MCS.2010.937003

[13] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan Su. 2019. *This* looks like *that*: Deep learning for interpretable image recognition. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS '19)*. 8928–8939. http://papers.nips.cc/paper/9095-this-looks-like-that-deep-learning-for-interpretable-image-recognition

[14] Dmitry Devetyarov and Ilia Nouretdinov. 2010. Prediction with confidence based on a random forest classifier. In *Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations*. 37–44.

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

[16] Souradeep Dutta, Xin Chen, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2019. Sherlock—A tool for verification of neural network feedback systems: Demo abstract. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '19)*. ACM, 262–263. https://doi.org/10.1145/3302504.3313351

[17] Souradeep Dutta, Yahan Yang, Elena Bernardis, Edgar Dobriban, and Insup Lee. 2022. Memory classifiers: Two-stage classification for robustness in machine learning. *arXiv:2206.05323* (2022). https://doi.org/10.48550/ARXIV.2206.05323

[18] Yeli Feng, Daniel Jun Xian Ng, and Arvind Easwaran. 2021. Improving variational autoencoder based out-of-distribution detection for embedded real-time applications. *ACM Transactions on Embedded Computing Systems* 20, 5s (2021), 1–26.

[19] Yeli Feng, Daniel Jun Xian Ng, and Arvind Easwaran. 2021. Improving variational autoencoder based out-of-distribution detection for embedded real-time applications. *ACM Transactions on Embedded Computing Systems* 20, 5s (Sept. 2021), Article 95, 26 pages. https://doi.org/10.1145/3477026

[20] R. A. Fisher. 1932. *Statistical Methods for Research Workers* (4th ed.). Oliver & Boyd.

[21] Daniel J. Fremont, Johnathan Chiu, Dragos D. Margineantu, Denis Osipychev, and Sanjit A. Seshia. 2020. Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI. In *Computer Aided Verification*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer International Publishing, Cham, 122–134.

[22] K. Fukunaga and P. M. Narendra. 1975. A branch and bound algorithm for computing *k*-nearest neighbors. *IEEE Transactions on Computers* C-24, 7 (1975), 750–753. https://doi.org/10.1109/T-C.1975.224297

[23] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP '18)*. 3–18. https://doi.org/10.1109/SP.2018.00058

[24] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[25] Radoslav Ivanov, Taylor J. Carpenter, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2020. Case study: Verifying the safety of an autonomous racing car with a neural network controller. In *Proceedings of the 23rd*

International Conference on Hybrid Systems: Computation and Control (HSCC '20). ACM, 1–7. https://doi.org/10.1145/3365365.3382216

[26] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2019. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '19)*. ACM, 169–178. https://doi.org/10.1145/3302504.3311806

[27] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Computer Aided Verification*. Lecture Notes in Computer Science, Vol. 10426. Springer, 97–117.

[28] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, David L. Dill, Mykel J. Kochenderfer, and Clark Barrett. 2019. The Marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification*. Lecture Notes in Computer Science, Vol. 11561. Springer, 443–452.

[29] Ramneet Kaur, Radoslav Ivanov, Matthew Cleaveland, Oleg Sokolsky, and Insup Lee. 2020. Assurance case patterns for cyber-physical systems with deep neural networks. In *Proceedings of the International Conference on Computer Safety, Reliability, and Security*. 82–97.

[30] Ramneet Kaur, Susmit Jha, Anirban Roy, Sangdon Park, Edgar Dobriban, Oleg Sokolsky, and Insup Lee. 2022. iDECODe: In-distribution equivariance for conformal out-of-distribution detection, association for the advancement of artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7104–7114.

[31] Ramneet Kaur, Susmit Jha, Anirban Roy, Sangdon Park, Oleg Sokolsky, and Insup Lee. 2021. Detecting OODs as datapoints with high uncertainty. *arXiv preprint arXiv:2108.06380* (2021).

[32] Ramneet Kaur, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. 2021. Are all outliers alike? On understanding the diversity of outliers for detecting OODs. *arXiv preprint arXiv:2103.12628* (2021).

[33] Ramneet Kaur, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. 2023. Predicting out-of-distribution performance of deep neural networks using model conformance. In *Proceedings of the 2023 IEEE International Conference on Assured Autonomy (ICAA '23)*. IEEE, 19–28.

[34] Ramneet Kaur, Xiayan Ji, Souradeep Dutta, Michele Caprio, Yahan Yang, Elena Bernardis, Oleg Sokolsky, and Insup Lee. 2023. Using semantic information for defining and detecting OOD inputs. *arXiv preprint arXiv:2302.11019* (2023).

[35] Ramneet Kaur, Yiannis Kantaros, Wenwen Si, James Weimer, and Insup Lee. 2023. Detection of adversarial physical attacks in time-series image data. *arXiv preprint arXiv:2304.13919* (2023).

[36] Ramneet Kaur, Kaustubh Sridhar, Sangdon Park, Yahan Yang, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. 2023. CODiT: Conformal out-of-distribution detection in time-series data for cyber-physical systems. In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*. 120–131.

[37] Rikard Laxhammar and Göran Falkman. 2011. Sequential conformal anomaly detection in trajectories based on Hausdorff distance. In *Proceedings of the 14th International Conference on Information Fusion*. IEEE, 1–8.

[38] Rikard Laxhammar and Göran Falkman. 2015. Inductive conformal anomaly detection for sequential detection of anomalous sub-trajectories. *Annals of Mathematics and Artificial Intelligence* 74, 1 (2015), 67–94.

[39] Juncheng Li, Frank R. Schmidt, and J. Zico Kolter. 2019. Adversarial camera stickers: A physical camera-based attack on deep learning systems. *CoRR abs/1904.00759* (2019). http://arxiv.org/abs/1904.00759

[40] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[41] David Macêdo, Tsang Ing Ren, Cleber Zanchettin, Adriano L. I. Oliveira, and Teresa Ludermir. 2021. Entropic out-of-distribution detection. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN '21)*. IEEE, 1–8.

[42] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. *arXiv:1312.5602 [cs.LG]* (2013).

[43] R. T. Ng and Jiawei Han. 2002. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering* 14, 5 (2002), 1003–1016. https://doi.org/10.1109/TKDE.2002.1033770

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[45] Ricardo Silva Peres, Xiaodong Jia, Jay Lee, Keyi Sun, Armando Walter Colombo, and Jose Barata. 2020. Industrial artificial intelligence in Industry 4.0: Systematic review, challenges and outlook. *IEEE Access* 8 (2020), 220121–220139. https://doi.org/10.1109/ACCESS.2020.3042874

[46] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified defenses against adversarial examples. *CoRR abs/1801.09344* (2018). http://arxiv.org/abs/1801.09344

[47] Shreyas Ramakrishna, Zahra Rahiminasab, Gabor Karsai, Arvind Easwaran, and Abhishek Dubey. 2021. Efficient out-of-distribution detection using latent space of β-VAE for cyber-physical systems. *arXiv preprint arXiv:2108.11800* (2021).

[48] Étienne Roquain. 2010. Type I error rate control for testing many hypotheses: A survey with proofs. *arXiv:1012.4078* (2010).

[49] Sriram Sankaranarayanan, Souradeep Dutta, and Sergio Mover. 2019. Reaching out towards fully verified autonomous systems. In *Reachability Problems*, Emmanuel Filiot, Raphaël Jungers, and Igor Potapov (Eds.). Springer International Publishing, Cham, 22–32.

[50] Erich Schubert and Peter J. Rousseeuw. 2018. Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms. *CoRR abs/1810.05691* (2018). http://arxiv.org/abs/1810.05691

[51] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489. https://doi.org/10.1038/nature16961

[52] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR abs/1312.6034* (2014).

[53] Kaustubh Sridhar, Souradeep Dutta, Ramneet Kaur, James Weimer, Oleg Sokolsky, and Insup Lee. 2022. Towards alternative techniques for improving adversarial robustness: Analysis of adversarial training at a spectrum of perturbations. *arXiv preprint arXiv:2206.06496* (2022).

[54] Vijaya Kumar Sundar, Shreyas Ramakrishna, Zahra Rahiminasab, Arvind Easwaran, and Abhishek Dubey. 2020. Out-of-distribution detection in multi-label datasets using latent space of β-VAE. In *Proceedings of the 2020 IEEE Security and Privacy Workshops (SPW '20)*. IEEE, 250–255.

[55] Sudharson Sundararajan, Ismail Zohdy, and Booz Allen Hamilton. 2016. *Vehicle Automation and Weather: Challenges and Opportunities*. U.S. Department of Transportation.

[56] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. 2020. CSI: Novelty detection via contrastive learning on distributionally shifted instances. *arXiv preprint arXiv:2007.08176* (2020).

[57] Ashish Tiwari, Bruno Dutertre, Dejan Jovanović, Thomas de Candia, Patrick D. Lincoln, John Rushby, Dorsa Sadigh, and Sanjit Seshia. 2014. Safety envelope for security. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*. 85–94.

[58] Paolo Toccaceli and Alexander Gammerman. 2017. Combination of conformal predictors for classification. *Proceedings of Machine Learning Research* 60 (2017) 39–61.

[59] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. Springer Science & Business Media.

[60] Vladimir Vovk, Ilia Nouretdinov, and Alexander Gammerman. 2003. Testing exchangeability on-line. In *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*. 768–775.

[61] Vladimir Vovk and Ruodu Wang. 2020. Combining p-values via averaging. *Biometrika* 107, 4 (2020), 791–808. https://doi.org/10.1093/biomet/asaa027

[62] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[63] Daniel Wilson. 2019. The harmonic mean p-value for combining dependent tests. *Proceedings of the National Academy of Sciences* 116 (2019), 201814092. https://doi.org/10.1073/pnas.1814092116

[64] Eric Wong and Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. *Proceedings of Machine Learning Research* 80 (2018), 5286–5295. https://proceedings.mlr.press/v80/wong18a.html

[65] Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. 2017. Reachable set computation and safety verification for neural networks with ReLU activations. *arXiv:1712.08163* (2017). https://doi.org/10.48550/ARXIV.1712.08163

[66] Yahan Yang, Ramneet Kaur, Suradeep Dutta, and Insup Lee. 2022. Interpretable detection of distribution shifts in learning enabled cyber-physical systems. In *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS '22)*.

[67] Ev Zisselman and Aviv Tamar. 2020. Deep residual flow for out of distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13994–14003.

Received 25 July 2022; revised 11 October 2023; accepted 17 January 2024