

Online Distribution Shift Detection via Recency Prediction

Rachel Luo, Rohan Sinha, Ali Hindy, Shengjia Zhao, Silvio Savarese, Edward Schmerling, Marco Pavone

Abstract—When deploying modern machine learning-enabled robotic systems in high-stakes applications, detecting distributional shift is critical. However, most existing methods for detecting distribution shift are not well-suited to robotics settings, where data often arrives in a streaming fashion and may be very high-dimensional. In this work, we present an online method for detecting distributional shift with guarantees on the false positive rate — i.e., when there is no distribution shift, our system is very unlikely (with probability $< \epsilon$) to falsely issue an alert; any alerts that are issued should therefore be heeded. Our method is specifically designed for efficient detection even with high dimensional data, and it empirically achieves up to 6x faster detection on realistic robotics settings compared to prior work while maintaining a low false negative rate in practice (whenever there is a distribution shift in our experiments, our method indeed emits an alert).

I. INTRODUCTION

Machine learning (ML) models deployed in the real world often encounter test time inputs that do not follow the same distribution as the training time inputs, because autonomous robots continuously encounter new situations when deployed — in other words, there is *distribution shift*. However, standard machine learning practice operates under the assumption that the training and test distributions are identical, and thus learned models may not perform well under changed conditions. Consequently, detecting distribution shift is very important for maintaining the reliability of modern ML-enabled systems, especially in high stakes situations such as aircraft control, autonomous driving, or medical decision-making. For instance, with an ML- or computer vision-assisted aircraft control system, the pilot should be alerted if the environment has changed drastically; in an autonomous driving setting, a human operator should be alerted if a sudden downpour changes the performance of the car’s automatic emergency braking system.

In robotics settings, data often arrives online in real-time, so detecting distribution shift in an online manner is particularly important: knowledge of distribution shifts can trigger safety-preserving interventions and subsequent model refinement or retraining. However, most existing methods for detecting a distribution shift operate only in an offline, batch setting. Such methods typically test whether two sets of samples originated from the same distribution, and are not easily adapted to the online setting most relevant in robotics. Moreover, when distributions drift gradually over time, it might be impossible to pinpoint when a meaningful change

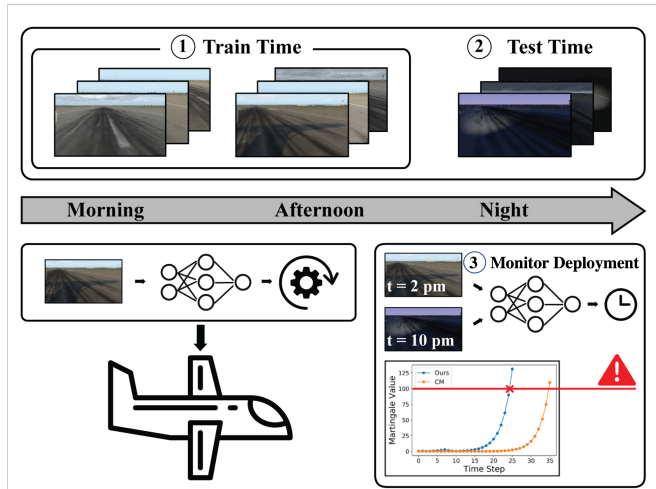


Fig. 1: Illustration of our problem setting and high-level approach. 1) Learning enabled robotics systems, such as a vision-based aircraft controller, are trained on data from a finite set of environments. 2) When deployed, these systems may operate in distribution-shifted conditions, resulting in erroneous predictions on out-of-distribution data. 3) To improve safety, we design a warning system that can detect distributional shifts in a streaming fashion with a *guaranteed* false positive rate. Our method tests for exchangeability of time-series data: We signal that a distributional shift has occurred if a neural network can consistently distinguish which samples are more recent.

has occurred. For example, camera degradation may slowly increase noise levels in images over time.

If a distribution shift has indeed occurred, it should be detected as soon as possible, since distribution shifts that have gone unnoticed can lead to undesirable or dangerous situations. For instance, a drone using a vision system trained in bright environments may crash in dark environments. An automatic emergency braking system trained with respect to the distribution of drivers in one particular city may be too slow to brake in a different city. However, any warning system that gives too many false positives will be ignored by the user, and therefore not useful in practice. Thus, a good warning system should issue alerts about distribution shifts very quickly, but give very few false positives.

In this work, we focus on episodic situations with gradually shifting distributions. In such situations, a warning system ideally issues an alert *before* a problem arises due to the magnitude of the distribution shift. As an example, for a plane repeatedly taxiing down a runway during a continuous deployment, each taxiing sequence can be considered an “episode” drawn from a task distribution. However, the plane’s sensors may degrade over time, or the outside lighting conditions may change significantly over the course of the day, and we would like a warning before these shifted

R. Luo, R. Sinha, A. Hindy, S. Zhao, S. Savarese, E. Schmerling, and M. Pavone are with Stanford University, Stanford, CA, USA; {rsluo, rhnsinha, ahindy, sjzhao, ssilvio, schmerling, pavone}@stanford.edu.

The NASA University Leadership Initiative (grant #80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity.

conditions cause a major problem.

We present a system designed to address the tri-fold challenge of 1) detecting distribution shift 2) in an online setting 3) quickly and with guarantees, for the gradually shifting episodic situations described above. Our online system quickly alerts users when a distribution shift has occurred, while also providing a guaranteed (low) false positive rate. We build on existing martingale methods to provide a guarantee on the number of false positives; i.e. when there is no distribution shift, no warning will be issued with at least $1 - \epsilon$ probability. This means that there will be very few extraneous warnings, so any warnings that are emitted should be heeded. In contrast with prior works, which are often inefficient for high dimensional data (e.g. images), we directly train a neural network model to predict whether a new sample at test time differs meaningfully from previous samples, and combine this learned model with a martingale to issue warnings with guarantees. This approach empirically leads to faster alerts.

In summary, the contributions of our paper are as follows: 1) We present a method that can detect distributional shifts on high-dimensional data in an online fashion, thereby improving upon existing methods that are either offline or not tailored for high-dimensional input data. 2) We construct a warning signal that grows exponentially under distribution shift, allowing us to detect online shifts more rapidly than existing approaches. 3) We empirically evaluate our approach on several standard synthetic benchmarks (CIFAR-100 [1], CIFAR-10 [2], and the Wine Quality dataset [3]), as well as on photorealistic simulations of an autonomous aircraft taxiing down a runway using a camera perception module in the X-Plane simulator. Our approach detects distributional shifts up to six times as rapidly as the baseline while maintaining a guaranteed false positive rate of 1%, demonstrating that our method performs well on realistic examples. We conclude that our method is attractive for robotic applications as it is practical and tailored to detect shifts in the high-dimensional and sequentially observed inputs of ML models, like perception systems, used in a robotic autonomy stack.

The rest of this paper will be organized as follows. In Sections II and III, we review related work and provide background information on martingales. In Section IV, we describe our proposed method, including the martingale that we construct. Finally, in Sections V and VI, we present experimental results on several synthetic datasets as well as on a photorealistic aircraft control dataset.

II. RELATED WORK

Machine learning models can perform poorly and erratically on test data drawn from a distribution that differs from the training distribution. Mitigating the impact of distributional shifts is a long-standing challenge, and empirical studies show that subtle shifts still severely impact the performance of state-of-the-art models (e.g., see [4], [5], [6], [7]). Furthermore, as learned components find increasing use in robotics stacks, erroneous predictions induced by distributional shifts can cause dangerous system-level failures in safety-critical applications. For example, a robot using a classification model trained on daytime data can cause

accidents when deployed at night. Therefore, we must develop methods to detect distributional shifts to avoid system failures in shifted conditions.

The problem of detecting distribution shift has long been studied by both the machine learning community and the statistics community. Traditional approaches use statistical hypothesis testing to determine whether the test-time distribution differs from the training distribution [8], [9], [10], [11]. For example, [8] develop a hypothesis test based on evaluating the maximum mean discrepancy (MMD) and similarly, [9] use a dimensionality reduction technique followed by a statistical two-sample test to compare the two distributions. [10] develop conditional distribution hypothesis tests and propose a score-based test statistic for localizing distribution shift. In robotics, [12] apply a two-sample procedure to detect when a robot is operating under shifted conditions that harm its performance. However, these methods are typically designed for an offline (batch) setting, and there is no obvious way to use these methods online without either losing the guarantee, or being very inefficient statistically.

Another approach for detecting distribution shift, introduced by [13], uses conformal martingales to test for exchangeability and is currently the only technique for detecting distribution shift online [14], [15], [16], [17], [18], [19], [20], [21]. These methods use conformal prediction to obtain p-values for each sample at test time, and then use these p-values to define a martingale. If the martingale grows large, then there has likely been a distribution shift. [15] is the most recent and most relevant to our work, as it combines conformal prediction with martingale theory to obtain an online distribution shift detector with a guarantee limiting the false positive rate. The authors compute a conformal p-value for each sample, and then use those p-values to define a Simple Jumper martingale. They experiment with several nonconformity score functions and find that the nearest distance nonconformity score performs best. Intuitively, they train a predictor and look at its performance, and then check whether the predictor’s performance degrades for new samples — if so, there has likely been a distribution shift. Their work demonstrates good efficiency on the Wine Quality dataset, which contains 11-dimensional feature vectors; i.e., they detect distribution shifts quickly.

However, these martingales generally do not perform well on more complex or higher-dimensional robotics settings (e.g. with image data), and they are not directly optimized to solve the problem of detecting distribution shift in an end-to-end manner. Additionally, these methods will only detect a distribution shift if the shift affects the specific predictor used to define the nonconformity score, which may be undesirable if there are other metrics that are also important, or if the overall predictor performance stays the same but the predictor now fails more often in more critical situations. Instead, we design a more efficient martingale based on a learned classifier; our martingale detects distribution shifts more quickly and does not have these drawbacks.

III. BACKGROUND

A martingale is a stochastic process (a sequence of random variables) where the conditional expectation of the next observation, given all previous observations, is the same

as the most recent observation. Many stochastic processes of interest are martingales, and therefore there is a well-developed body of statistical theory on martingales that we can draw from [22], [23], [24], [25].

Definition 1 (Martingale) A martingale is a sequence of random variables M_1, M_2, \dots , such that $E[M_n] < \infty$ for all n , and

$$E[M_{n+1} | M_1, \dots, M_n] = M_n. \quad (1)$$

A martingale can be thought of as the amount of capital of a player who participates in a series of fair bets — regardless of the historical observations, the player’s expected capital at any point in the future is the same as his current capital. Thus, in expectation, he neither wins nor loses any money. In fact, the probability that a martingale grows very large (i.e. the player wins a lot of money from these fair games) is very low. Doob’s Inequality formalizes this notion.

Proposition 1 (Doob’s Inequality) For a martingale M_n indexed by an interval $[0, N]$, and for any positive real number C , it holds that

$$\Pr \left[\sup_{0 \leq n \leq N} M_n \geq C \right] \leq \frac{E[\max(M_N, 0)]}{C}. \quad (2)$$

In other words, the probability that the martingale *ever* grows larger than C is inversely proportional to C .

In our work, we define a martingale M_n based on the outputs of a trained predictive model and apply Doob’s Inequality to obtain probabilistic guarantees bounding the false positive rate. However, the M_n that we define is a martingale only if new data points observed at test time are *exchangeable* with data points seen during training.

Definition 2 (Exchangeability) A sequence of data points X_1, X_2, \dots, X_N is exchangeable if the probability of observing any permutation of X_1, X_2, \dots, X_N is equally likely.

Under the hypothesis of exchangeability, the probability of M_n growing large is small. In other words, if there is no distribution shift (the data points observed during training and after deployment are exchangeable), then the probability that our system falsely issues a warning (M_n grows large) is small. Conversely, if the martingale grows large, then the data was likely not exchangeable, implying that a distribution shift occurred.

IV. DETECTING DISTRIBUTION SHIFT

We propose a method for detecting distribution shift online in episodic robotics settings. Our method combines a learned, end-to-end approach with statistical martingale theory to issue alerts about distribution shifts with a guaranteed false positive rate.

A. Problem Setup

Let $D_{\text{orig}} = \{X_1, X_2, \dots, X_n\}$ be a sequence of past data points, where each point represents an episode of the robot executing in some environment, and let $D_{\text{new}} = X'_1, X'_2, \dots$ be a sequence of new data points observed at test time, where each point again represents an episode. (In our experiments,

each data point is simply a random sample from the episode.) Our goal is to determine in an online manner whether these more recent points X'_j are drawn from the same distribution as the original points X_i . More formally, we aim to design a series of test functions

$$\psi_j : D_{\text{orig}}, X'_1, \dots, X'_j \mapsto \{T, F\}, \forall j = 1, 2, \dots, \quad (3)$$

where the output $T(\text{true})$ indicates that we have found a distribution shift, and $F(\text{false})$ indicates that we have not.

We say that the test is ε -sound if whenever there is no distribution shift, i.e., when the test data X'_1, X'_2, \dots are indeed exchangeable with D_{orig} then

$$\Pr[\exists j, \psi_j(D_{\text{orig}}, X'_1, \dots, X'_j) = T] \leq \varepsilon \quad (4)$$

and this guarantee should hold for any distributions of D_{orig} and test data X'_1, X'_2, \dots . Intuitively, a test is sound if whenever there is no distribution drift, our test never issues a warning with high $1 - \varepsilon$ probability.

Conversely, when there is a distribution shift, we want the test to issue a warning as soon as possible; i.e. we want a small j such that ψ_j outputs $T(\text{true})$. Formally, we define the initial discovery time as the smallest j such that ψ_j is $T(\text{true})$. While we will show that it is possible to guarantee soundness for *any* data distributions, it is generally impossible to guarantee the initial discovery time (unless the test trivially issues a warning all the time). For example, in the case where the distribution shift is tiny, e.g. the total variation distance between X'_1, X'_2, \dots and the initial data D_{orig} is very small, there are fundamental lower bounds on how well a test can distinguish the two distributions [26].

In this paper, we will devise a test that guarantees ε -soundness, and has low initial discovery time empirically.

B. Proposed Method

The key idea behind our method is that a predictor trained to distinguish between two samples, one of which is taken from D_{new} and the other of which is taken from D_{orig} , can do no better than random chance if there has been no distribution shift. That is, an indicator variable Y_k that takes the value of 1 when the prediction model correctly predicts which sample originated from D_{new} and 0 otherwise is a Bernoulli random variable with parameter $p := \Pr[Y_k = 1] = 0.5$ when no distribution shift has occurred. This is true no matter what the prediction model is, or how it was trained. We formalize this notion in Section VIII-A of the Appendix of the extended version of this paper [27].

More formally, let $X_i \in \mathcal{X}$ and $X'_j \in \mathcal{X}'$ represent samples from D_{orig} and D_{new} respectively, and let $f : \mathcal{X} \times \mathcal{X}' \rightarrow \{0, 1\}$ be a trained neural network model that takes in as input a pair of unordered, unlabeled samples (X_i, X'_j) , and predicts which input sample is the more recent of the two (i.e. which is from D_{new}). Note that f is a binary classifier. At each time step k , we can then define an indicator variable Y_k as follows:

$$Y_k = \begin{cases} 1 & \text{if } f \text{ predicts correctly} \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Y_k is a Bernoulli random variable with $p = 0.5$ if no distribution shift has occurred.

Then, we use these Y_k values to define a stochastic process M_n , which is a martingale under the hypothesis that there is no distribution shift. Therefore, as per Doob's Inequality, M_n will not grow too large with high probability if there is no distribution shift. If the martingale does grow large, then the assumption that our indicators are Bernoulli random variables with $p = 0.5$ has most likely been violated, which means that the samples are not exchangeable and that a distribution shift has therefore occurred with high probability.

Thus, we seek to train a model whose recency predictions are better than random chance. We do this by training a neural network that takes in as input two unordered, unlabeled samples (the most recent sample from the test data points X'_j , and a random sample from the past data points X_i), and predicts which is the more recent sample. If the samples are indeed exchangeable, it will be impossible to do better than a Bernoulli RV with $p = 0.5$, regardless of how well we train our model. If the samples are not exchangeable, our model should do well and the martingale will grow. This makes intuitive sense — if the predictor accuracy is e.g. 100% (i.e. it is very easy to distinguish the samples from D_{orig} and D_{new}), there must have been a distribution shift. We can set a threshold C based on Doob's Inequality such that if the martingale grows larger than C , it is highly unlikely (with probability proportional to $1/C$) that there was no distribution shift.

Note that our proposed method is self-supervised: it works even when the input data is unlabeled. This contrasts with the method in [15], which looks at the performance of a trained predictor on each training sample and observes whether the predictive performance degrades for new samples at test time. Instead, our method instead directly trains a model to predict whether a new sample differs from the older samples, since it can be difficult or expensive to obtain ground-truth labels online. Thus, compared to the method in [15], ours is end-to-end, directly detects differences, and uses deep learning. Compared to classical ML methods, ours can be deployed in an online manner since we use a martingale rather than a two-sample test.

C. Choice of Martingale

In theory, any martingale constructed from the indicators Y_k in (5) would allow us to detect distribution shift, in the sense that the martingale would eventually grow large if Y_k is not actually Bernoulli with $p = 0.5$. However, one desirable property for our martingale is that it should grow quickly if there has been a distribution shift (the Y_k are not Bernoulli with $p = 0.5$). Thus, we use an exponential martingale defined as follows:

$$M_n = e^{t \cdot S_n} / (q + pe^t)^n. \quad (6)$$

where $S_n = \sum_{k=1}^n Y_k$, and $p = q = 0.5$. We prove in Appendix VIII-A that M_n is indeed a martingale.

Since $M_0 = 1$ and the martingale is non-negative, Doob's Inequality simplifies in this case to

$$\Pr \left[\sup_{0 \leq n \leq N} M_n \geq C \right] \leq \frac{1}{C}.$$

For our experiments, we use a threshold of $C = 100$, which guarantees a false positive rate of ≤ 0.01 .

D. Training Procedure

During training time, we observe a sequence of data points D_{orig} , and we can divide these into a held back set of “unseen” data points (which we will not use until test time), a set of “older” data points (i.e. the points from earlier in the sequence), and a set of “more recent” data points (the points from later in the sequence). (For our experiments, we divided the samples in D_{orig} into three approximately equally sized subsets.) We then take pairs of randomly selected samples (one from the subset of “older” data points and one from the subset of “more recent” data points), and train a neural network to distinguish between the two. In other words, the input to this neural network model is a pair of shuffled, randomly selected samples, and the output is either 0 or 1, depending on which sample is from the subset of more recent data points. Note that this is a self-supervised method — it depends only on the ordering of the samples.

At test time, we observe a sequence of data points D_{new} . Each incoming data point X'_j is paired with a randomly selected data point X_i from the subset of unseen data points from D_{orig} . This pair of samples is then input into the trained model, which makes a prediction. The output of the predictor is used to define an indicator variable according to Eq. 5, and a martingale according to Eq. 6. If $M_n > C$, an alert will be issued. Since the left hand side of Prop. 1 takes the supremum of M_n over n , the test function $\psi_n := \mathbb{1}\{M_n > C\}$, which issues an alert the first time the martingale is greater than C , will have a guaranteed false positive rate of $1/C$.

After the process described in the previous paragraph is completed for a data point X'_j in D_{new} , X'_j should be added to the subset of more recent data points from D_{orig} . Then, the entire process (including taking pairs of randomly selected samples and training the binary predictor model) will be repeated for X'_{j+1} . This step is necessary for detecting distribution shifts that occur during test time and have never been previously encountered during training.

V. SYNTHETIC EXPERIMENTS

We empirically validate our method on several synthetic datasets, and find that it consistently outperforms prior work.

Datasets. We use the CIFAR-100 [1], CIFAR-10 [2], and Wine Quality [3] datasets, which are standard benchmarks for existing work on distribution shift ([15] evaluated their method on the Wine Quality dataset). The CIFAR-100 and CIFAR-10 datasets consist of $32 \times 32 \times 3$ images. To simulate various distribution shifts on the CIFAR datasets, we use the CIFAR-100-C and CIFAR-10-C datasets [4], which are perturbed versions of the CIFAR test sets. Each dataset includes 15 perturbed versions of the original test set, with perturbations such as Gaussian noise, motion blur, and pixelate. The Wine Quality dataset comprises 11-dimensional feature vectors for 4898 white and 1599 red wines.

Experimental Setup. We compare our method against the method described by Vovk et. al. in [15], which we will refer to as the conformal martingale (CM) method. For the CIFAR experiments, only unperturbed images are used during training. At test time, the first ten images are unperturbed, and the remaining images are perturbed. For

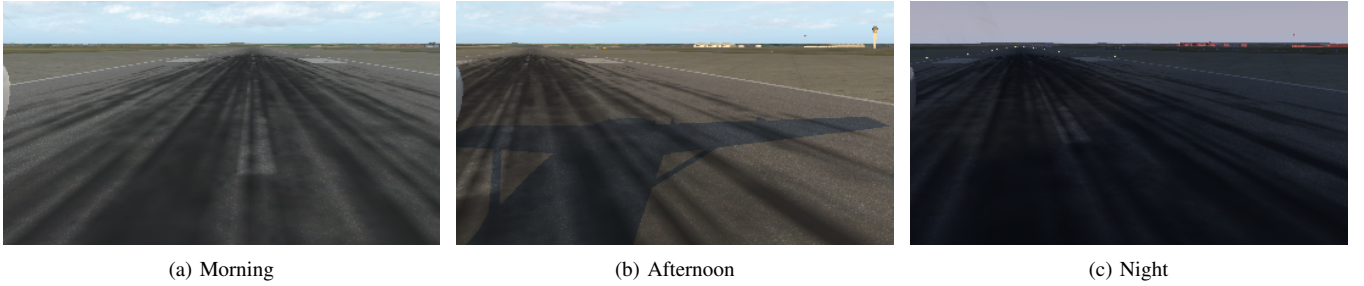


Fig. 2: Sample images generated with the X-Plane 11 flight simulator, from the (2a) morning, (2b) afternoon, and (2c) night. There is a distributional shift caused by gradually changing lighting conditions over the course of the day.

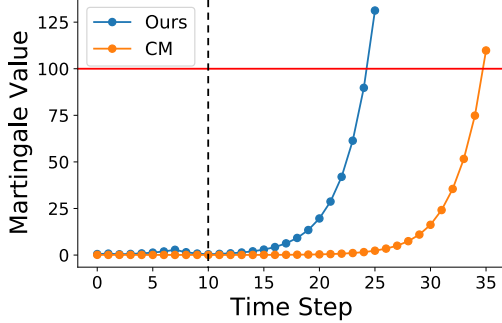


Fig. 3: Martingale values for **our method** (blue) and the **CM method** (orange). An alert is issued when the martingales reach the threshold of 100. Our method issues an alert at time step 26 (i.e. 16 time steps after the distribution shift), while the CM method issues an alert at time step 36 (26 time steps after the distribution shift). Note that the distribution shift occurs at time step 10.

the Wine Quality experiment, white wines are used during training and red wines are used at test time.

Results. Our method consistently outperforms the CM method. Over the 15 perturbations of CIFAR-100, our method takes an average of 24.25 time steps after the distribution shift to issue an alert, while the CM method takes an average of 71.33 time steps. For CIFAR-10, our method takes on average 22.72 time steps, while the CM method takes on average 56.77 time steps. The tables in Appendix VIII-B summarize the results for different perturbations of CIFAR-100 and CIFAR-10. In Fig. 3, we show an example plot for CIFAR-10 with the “pixelate” perturbation, demonstrating the martingale growth for both our method and the CM method. Our martingale grows more rapidly and issues an alert in fewer time steps. For the Wine Quality dataset, our method takes 18 time steps to issue an alert, while the CM method takes 24 time steps.

VI. X-PLANE SIMULATOR EXPERIMENTS

Finally, we validate the performance of our method on image data from an autonomous aircraft that relies on an outboard camera in a photorealistic flight simulator. The autonomous aircraft uses a PID controller to taxi along the centerline of a runway. We induce two separate distribution shifts, both of which cause the autonomous aircraft to fail and run off the runway. We use our method to detect these realistic failure scenarios of learning-enabled robots and find that it significantly outperforms prior work, detecting gradual distribution shifts up to 6x faster than the baselines. We also

show empirically that when there is no distribution shift, our martingale does not grow.

A. Gradual Daytime to Nighttime Shift

We first demonstrate that our method can efficiently detect distributional shifts by simulating a gradual daytime to nighttime shift. In our simulations, the lighting conditions gradually change over the course of the day.

Dataset. We use the X-Plane 11 flight simulator and NASA’s XPlaneConnect Python API to create 1000 simulated video sequences taken from a camera attached to the outside of the plane as it taxis down the runway at different times throughout the day (with different weather conditions, starting positions, etc.) [28]. The first 295 sequences take place in the morning (8:00am-12:00pm), the next 344 sequences take place in the afternoon (12:00pm-5:00pm), and the last 361 sequences take place at night (5:00pm-10:00pm). Each taxiing sequence consists of approximately 30 images of size 200x360x3 (note that these images are much larger than those in the CIFAR dataset). We randomly sample one image from each sequence. Fig. 2 shows an example of an image from the morning, afternoon, and night.

Methods. We compare our method against two baselines. The first is the CM method as described in [15], using a nearest distance nonconformity score. As a second baseline, we add deep learning to the CM method (CM+DL) to disentangle the effect of using a learned neural network model from the change in the methodology itself. Here, we use the confidence output by a trained neural network model as the nonconformity score.

Experimental Setup. We combine the morning and afternoon data points to form the training dataset with a total of 639 data points. These are then divided into 213 “unseen” images (to pair with the test time images), and 213 image pairs for training the neural network model. The evening data points are deployed in time order (i.e. the earliest evening images first) at test time. We train a basic four layer convolutional neural network to predict which inputs are more recent.

Results. Our method significantly outperforms both baselines. Our method issues an alert only 13 time steps into the evening data samples. With the CM method, the alert is issued only after 80 time steps, and with the CM+DL method, the alert is issued after 52 time steps. Fig. 5 plots the growth of the martingale values for each method; an alert is issued after each martingale crosses the threshold of 100. The prompt alert from our method is particularly interesting

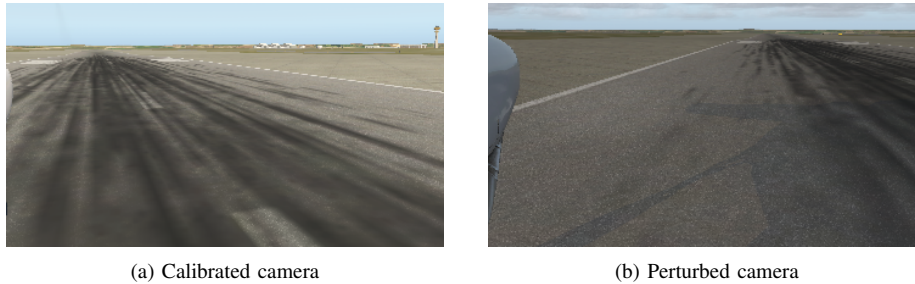


Fig. 4: Sample images generated with the X-Plane 11 flight simulator, with (4a) a standard camera angle, and (4b) a perturbed camera angle. The standard camera angle might represent a calibrated camera, while the perturbed camera angle might represent a camera that has been knocked slightly askew.

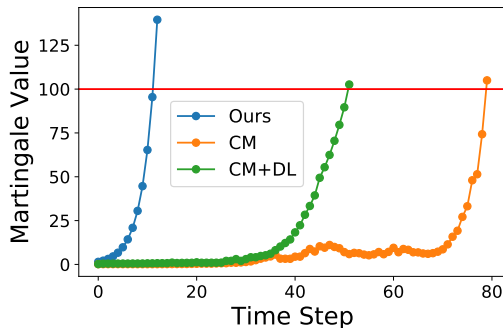


Fig. 5: Martingale values for **our method** (blue), the **CM method** (orange), and the **CM+DL method** (green). An alert is issued when the martingales reach 100. Our method issues an alert at time step 13, CM+DL issues an alert at time step 52, and CM issues an alert at time step 80.

because the early evening images (from just after 5:00pm) look visually very similar to those from earlier in the day.

These results indicate that our method performs well on realistic examples, and detects distributional shifts up to 6x more quickly than prior work. They also suggest that our method holds a larger efficiency advantage as the data increases in dimensionality, and that both our end-to-end optimized methodology and our inclusion of a deep learning model lead to more rapid detection of distributional shifts.

B. Camera Angle Shift

We also demonstrate that when there is no distribution shift, our martingale does not grow large. In this set of simulations, we compare the growth of our martingale with and without a distribution shift, where the distribution shift is a change in the camera angle.

Dataset. We again use the X-Plane 11 flight simulator to create 600 video sequences taken from a camera attached to the outside of the plane as it taxis down the runway [28]. These sequences occur at randomly initialized times between 8:00am and 10:00pm. Of these 600 sequences, 400 are taken with a standard camera angle, and 200 are taken with a slightly perturbed camera angle (see Fig. 4), simulating the camera being knocked slightly askew. Each taxiing sequence consists of approximately 30 images of size 200x360x3, and we randomly sample one image from each sequence.

Experimental Setup. At training time, we observe 200 samples with the standard camera angle. At test time, we observe either 200 samples with the perturbed camera angle

(a distribution shift), or 200 different samples with the standard camera angle (no distribution shift). This experiment can be thought of as simulating a calibrated camera setup in the standard case, and a camera that has been knocked slightly askew in the perturbed case.

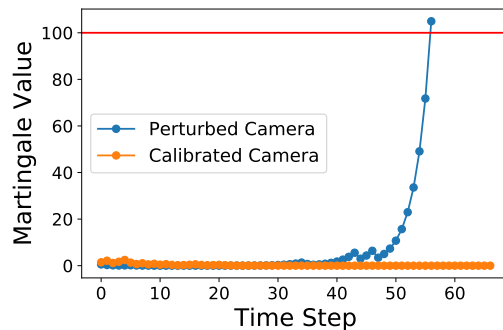


Fig. 6: Martingale values for our method **with** (blue) and **without** (orange) distribution shift. In this case, the distributional shift is caused by a perturbation in the camera angle. With a distribution shift, the martingale grows rapidly, but without one, the martingale does not grow.

Results. The results for this experiment are shown in Fig. 6, where “Perturbed Camera” is the distribution shift case and “Calibrated Camera” is the no distribution shift case. When there is no distribution shift, the martingale does not grow large; when there is a distribution shift, the martingale grows quickly.

VII. CONCLUSION

In this work, we introduce a method for detecting distributional shifts on high-dimensional data in a streaming fashion. Our method is practical for robotics applications, and we demonstrate empirically that it performs well on photorealistic simulations of a plane taxiing down a runway — it detects distributional shifts up to 6x more quickly than prior work. A limitation of our work is that it applies specifically to episodic settings, where the episodes can be considered exchangeable — our method cannot provide guarantees for points sampled sequentially from within the same episode, since they may be highly correlated. In future work, we would like to explore methods for detecting distributional shifts when the data is correlated.

REFERENCES

- [1] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-100 (canadian institute for advanced research),”
- [2] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research),”
- [3] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” *Decision Support Systems*, 2009.
- [4] D. Hendrycks and T. G. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” *ICLR*, 2019.
- [5] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, “Shortcut learning in deep neural networks,” *Nature Machine Intelligence*, vol. 2, pp. 665–673, Nov 2020.
- [6] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee, E. David, I. Stavness, W. Guo, B. Earnshaw, I. Haque, S. M. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang, “Wilds: A benchmark of in-the-wild distribution shifts,” in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664, PMLR, 18–24 Jul 2021.
- [7] J. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt, “Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization,” 2021.
- [8] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *J. Mach. Learn. Res.*, vol. 13, p. 723–773, mar 2012.
- [9] S. Rabanser, S. Günnemann, and Z. C. Lipton, “Failing loudly: An empirical study of methods for detecting dataset shift,” in *NeurIPS*, 2019.
- [10] S. M. Kulinski, S. Bagchi, and D. I. Inouye, “Feature shift detection: Localizing which features have shifted via conditional distribution tests,” *ArXiv*, vol. abs/2107.06929, 2020.
- [11] V. M. Kamulete, “Test for non-negligible adverse shifts,” *ArXiv*, vol. abs/2107.02990, 2021.
- [12] A. Farid, S. Veer, and A. Majumdar, “Task-driven out-of-distribution detection with statistical guarantees for robot learning,” in *Proceedings of the 5th Conference on Robot Learning*, vol. 164 of *Proceedings of Machine Learning Research*, pp. 970–980, PMLR, 08–11 Nov 2022.
- [13] V. Vovk, I. Nourtdinov, and A. Gammerman, “Testing exchangeability on-line,” in *ICML*, 2003.
- [14] V. Vovk, “Testing randomness online,” *Statistical Science*, 2021.
- [15] V. Vovk, I. Petej, I. Nourtdinov, E. A. Helgee, L. Carlsson, and A. Gammerman, “Retrain or not retrain: Conformal test martingales for change-point detection,” in *COPA*, 2021.
- [16] C. Eliades and H. Papadopoulos, “A histogram based betting function for conformal martingales,” in *COPA*, 2020.
- [17] D. Volkhonskiy, E. Burnaev, I. Nourtdinov, A. Gammerman, and V. Vovk, “Inductive conformal martingales for change-point detection,” in *COPA*, 2017.
- [18] V. Fedorova, A. Gammerman, I. Nourtdinov, and V. Vovk, “Plug-in martingales for testing exchangeability on-line,” *ArXiv*, vol. abs/1204.3251, 2012.
- [19] S. S. Ho, “A martingale framework for concept change detection in time-varying data streams,” *Proceedings of the 22nd international conference on Machine learning*, 2005.
- [20] A. Podkopaev and A. Ramdas, “Tracking the risk of a deployed model and detecting harmful distribution shifts,” 2021.
- [21] X. Hu and J. Lei, “A distribution-free test of covariate shift using conformal prediction,” 2020.
- [22] J. L. Doob, “What is a martingale?,” *The American Mathematical Monthly*, vol. 78, no. 5, pp. 451–463, 1971.
- [23] P. Hall and C. C. Heyde, *Martingale limit theory and its application*. Academic press, 2014.
- [24] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.
- [25] G. Shafer and V. Vovk, *Game-Theoretic Foundations for Probability and Finance*, vol. 455. John Wiley & Sons, 2019.
- [26] B. Yu, “Assouad, fano, and le cam,” in *Festschrift for Lucien Le Cam*, pp. 423–435, Springer, 1997.
- [27] R. Luo, R. Sinha, A. Hindy, S. Zhao, S. Savarese, E. Schmerling, and M. Pavone, “Online distribution shift detection via recency prediction.” <http://asl.stanford.edu/wp-content/papercite-data/pdf/Luo.Sinha.ICRA23.pdf>, 2022.
- [28] S. M. Katz, A. Corso, S. Chinchali, A. Elhafsi, A. Sharma, M. J. Kochenderfer, and M. Pavone, “Nasa uli xplane simulator.” https://github.com/StanfordASL/NASA_ULI_Xplane_Simulator, 2021.

VIII. APPENDIX

A. Proofs

Lemma 1 Let $\{Y_1, Y_2, \dots\}$ be a sequence of independent and identically distributed Bernoulli random variables with $\Pr(Y_i = 1) = p$, and define $S_n := \sum_{i=1}^n Y_i$. Then the stochastic process $\{M_n\}_{n=1}^\infty$, with

$$M_n = \frac{e^{t \cdot S_n}}{((1-p) + pe^t)^n}$$

is a Martingale.

Proof: First, note that $S_n \leq n$ for all n . Therefore, it holds that $|M_n| = M_n \leq \infty$ for all $n \geq 0$, so $\{M_n\}$ is integrable. Secondly, we verify that M_n satisfies equation (1):

$$\begin{aligned} \mathbb{E}[M_{n+1} | M_1, \dots, M_n] &= \mathbb{E}\left[\frac{e^{tY_{n+1}}}{(1-p) + pe^t} M_n \mid M_1, \dots, M_n\right] && \text{(Def. of } M_n) \\ &= \mathbb{E}\left[\frac{e^{tY_{n+1}}}{(1-p) + pe^t} \mid M_1, \dots, M_n\right] M_n && \text{(Take out what is known)} \\ &= \mathbb{E}\left[\frac{e^{tY_{n+1}}}{(1-p) + pe^t}\right] M_n && (Y_{n+1} \text{ is indep. of } Y_1, \dots, Y_n) \\ &= \frac{(1-p) + pe^t}{(1-p) + pe^t} M_n = M_n && \text{(Evaluate the expectation)} \end{aligned}$$

Therefore, M_n is a Martingale. ■

Lemma 2 Let $\{X_1, X_2, \dots\}$ be a sequence of data points with $X_i \in \mathcal{X}$ and let $f : \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$ be a model that predicts which input was more recent. That is, for any pair of observations (x, x') , the prediction $f(x, x')$ represents the probability that x occurred before x' . Suppose we randomly sample any two data points (X_i, X_j) and predict which sample is more recent (i.e., occurred later) using a fixed threshold $c \in [0, 1]$. That is, we predict X_i is more recent than X_j if $f(X_i, X_j) \geq c$. If the sequence $\{X_1, X_2, \dots\}$ is exchangeable, then the probability P_c that f correctly predicts which point was more recent is

$$P_c := \mathbb{P}(\{f(X_i, X_j) \geq c, i > j\} \cup \{f(X_i, X_j) < c, i < j\}) = \frac{1}{2}. \quad (7)$$

Proof: Since the sequence $\{X_1, X_2, \dots\}$ is exchangeable, the events $i < j$ and $i > j$ are equally likely. Since $f \in [0, 1]$, $f(X_i, X_j)$ must be either greater than or equal to c , or less than c . Thus, it follows that

$$\begin{aligned} P_c &= \mathbb{P}(\{f(X_i, X_j) \geq c, i > j\} \cup \{f(X_i, X_j) < c, i < j\}) \\ &= \mathbb{P}(\{f(X_i, X_j) \geq c\}) \mathbb{1}\{i > j\} + \mathbb{P}(\{f(X_i, X_j) < c\}) \mathbb{1}\{i < j\}) \\ &= \mathbb{P}(\{f(X_i, X_j) \geq c\}) \cdot \frac{1}{2} + \mathbb{P}(\{f(X_i, X_j) < c\}) \cdot \frac{1}{2} \\ &= \frac{1}{2}. \end{aligned}$$

Therefore, the probability of correctly classifying which data point is more recent must be $1/2$, regardless of the choice of classifier. ■

B. Additional Experimental Results

We ran 100 trials of each CIFAR experiment (described in Section V) for every perturbation in the CIFAR-100-C and CIFAR-10-C datasets, and averaged the results. For the CM method, we used the nearest distance nonconformity score, as recommended in [15]. We also used a nearest distance nonconformity score on lower dimensional feature vectors extracted from a ResNet pretrained on CIFAR data (denoted as CM-FV). These results, comparing the number of time steps required before an alert is issued for our method and for the CM method, are summarized in Tables I and II.

Time Steps Needed for Alert CIFAR-10			
Perturbation	Ours	CM	CM-FV
Brightness	9.29	166.61	160.67
Contrast	12.66	23.52	125.46
Defocus Blur	27.32	104.86	92.27
Elastic Transform	32.99	180.63	33.08
Fog	15.66	29.33	70.96
Frost	13.72	61.9	40.47
Gaussian Noise	24.04	91.65	29.39
Glass Blur	31.03	–	20.54
Impulse Noise	20.66	37.41	15.10
Jpeg Compression	30.21	–	37.13
Motion Blur	25.12	78.34	60.06
Pixelate	29.17	–	33.63
Shot Noise	26.35	86.56	20.92
Snow	11.13	167.27	42.04
Zoom Blur	31.39	103.46	69.79
Overall	22.72	94.30	56.77

TABLE I: Number of time steps after a distribution shift occurs before an alert is issued, for our method, the CM method, and the CM method with lower-dimensional feature vectors (lower is better, best results shown in **bold**). These experiments are run under various types of perturbations on the CIFAR-10 dataset. Note that the CM method fails to detect a distribution shift within 500 samples for 3 of the perturbations. Our method significantly outperforms the CM method.

Time Steps Needed for Alert CIFAR-100			
Method	Ours	CM	CM-FV
Brightness	9.77	–	189.0
Contrast	13.75	22.92	129.43
Defocus Blur	27.17	121.10	95.74
Elastic Transform	38.34	180.5	55.34
Fog	15.92	28.02	63.52
Frost	11.75	52.88	56.39
Gaussian Noise	27.94	89.10	31.29
Glass Blur	33.31	–	36.24
Impulse Noise	20.62	37.0	27.7
Jpeg Compression	34.52	–	40.28
Motion Blur	28.32	98.98	96.4
Pixelate	30.82	–	58.47
Shot Noise	27.22	94.52	29.12
Snow	12.61	–	65.78
Zoom Blur	31.70	127.98	95.31
Overall	24.25	85.3	71.33

TABLE II: Number of time steps after a distribution shift occurs before an alert is issued, for our method, the CM method, and the CM method with lower-dimensional feature vectors (lower is better, best results shown in **bold**). These experiments are run under various types of perturbations on the CIFAR-100 dataset. Note that the CM method fails to detect a distribution shift within 500 samples for 4 of the perturbations. Our method significantly outperforms the CM method.