Towards Anomaly Detectors that Learn Continuously

Andrea Stocco† and Paolo Tonella†
†Software Institute - USI, Lugano, Switzerland
Email: {andrea.stocco, paolo.tonella}@usi.ch

***Abstract*—In this paper, we first discuss the challenges of adapting an already trained DNN-based anomaly detector with knowledge mined during the execution of the main system. Then, we present a framework for the continual learning of anomaly detectors, which records in-field behavioural data to determine what data are appropriate for adaptation. We evaluated our framework to improve an anomaly detector taken from the literature, in the context of misbehavior prediction for self-driving cars. Our results show that our solution can reduce the false positive rate by a large margin and adapt to nominal behaviour changes while maintaining the original anomaly detection capability.**

***Index Terms*—AI Testing, Anomaly Detection, Autonomous Driving Systems, Continual Learning**

I. Introduction

System-of-Systems are composed of independent and complex heterogeneous systems which collaborate to create capabilities not achievable by an individual system. Nowadays, an increasing number of such systems embed one or more Deep Neural Networks (DNNs) to perform tasks that cannot be expressed with traditional software logic [1]. Such software infrastructures perform advanced safety-critical tasks, such as autonomous driving [2], medical diagnosis [3], disease prediction [4], and aircraft collision avoidance [5].

Despite the successful results, DNNs carry numerous drawbacks when exposed to open world operational domains. For example, DNNs are fragile to domain shifts [6] (i.e., test data that differ from the training distribution) and data corruption [7] (e.g., adversarial examples) that may occur when the system is in operation. To date, DNN models are mostly tested relying on accuracy metrics on large test sets that attempt to represent a meaningful subset of the targeted operational domain. This is insufficient when the DNN is deployed within a real-world system as the examples gathered in the field may differ substantially from those of the test set.

In the machine learning literature, this problem is called out-of-distribution detection [8], whereas in the software testing literature it is mostly referred to as input validation [9]. A popular solution to address this problem is using unsupervised anomaly detection techniques that need no a priori knowledge of the anomalies, i.e., without a labelled dataset [10]. Rather, anomaly detectors identify the data that differ from the norm and that do not belong to the nominal data distribution. However, their effectiveness is limited by the representativeness of the data used to train them.

In this paper, we first discuss some of the challenges to build an anomaly detector that learns continuously from data gathered during the operation of the main system. Then, we propose an anomaly detection framework that not only monitors the main system, but also continually learns from in-field data, and, when the input data stream drifts from the available nominal data, re-trains the anomaly detector.

We instantiated our framework in the context of a real-world application domain, i.e., misbehaviour prediction for autonomous driving systems, to improve an autoencoder-based anomaly detector from the literature. Such an anomaly detector was used to anticipate misbehaviours of the main driving component within a simulation environment, by recognizing unexpected conditions such as weather changes.

In safety critical conditions, the true alarm rate (true positives) must be very close to 1, because unsafe executions should be eliminated or reduced to a negligible probability. However, such predictors can achieve a high true alarm rate only at the price of accepting a high false alarm rate (false positives) that can occur also in nominal or nearly-nominal conditions, causing much driver's discomfort and negatively affecting the driving experience.

On the other hand, an invaluable opportunity for improving this scenario is through the availability of unlabelled nominal data collected from the field. While such data is useless for retraining the main driving component, because it is unlabelled, it can be still used to improve the misbehaviour predictor, since no labels are required for its training.

The main challenge consists in carefully selecting the frames that can be used for adaptation. Our framework automates this task by comparing the score of the misbehaviour predictor with an in-field driving quality metric. If the misbehaviour predictor raises a false alarm (i.e., the driving scenario is unseen, but the driving component is indeed confident), our framework stores them in a buffer for adaptation, which is then used for retraining.

The main contributions of our work are:

- A framework for continual learning of DNN-based anomaly detectors that perform misbehaviour prediction.
- An instantiation and evaluation of our framework to improve an existing anomaly detector [11] under a diverse set of in- and out-of-distribution datasets. We show that our approach can reduce the false alarm rate by a large margin, without affecting the anomaly detection capability, i.e., the true alarm rate.

II. Background

A. Anomaly Detection and Autoencoders

Anomaly detection techniques have been used within a multitude of application domains such as intrusion detection systems [12], fraud detection [13], and IoT [14].

An *anomaly* can be defined as an observation that significantly deviates from other observations so as to arouse suspicion that it was generated by a different mechanism [15]. Anomalies can be caused by errors in data, but they can be also indicative of new, previously unknown, underlying scenarios.

In many real world domains, such as autonomous driving, abnormal data represent rare and unexpected events, for which no prior knowledge, or label, is available (hence, the context is *unsupervised*). Thus, research has moved to models that can be trained using no supervision, including one class SVM [16], clustering [17] and self-organizing maps [18]. In the spectrum of unsupervised anomaly detection solutions, architectures based on *autoencoders* (AEs) have emerged as a popular and very effective technique [10]. AEs are DNNs designed to reconstruct the inputs they are given. When trained on nominal data, AEs learn how to reconstruct normal data patterns, whereas they worsen their reconstruction capability when unknown inputs are given. Hence, nominal and anomalous inputs are distinguished by selecting an appropriate threshold based on the reconstruction errors obtained on a validation set.

B. Autoencoder-based Monitors

An accurate detection of hazardous situations is a necessary prerequisite for the implementation of a fail-safe system having a redundant component that analyzes the data fed to the main system and, in the face of unsupported inputs, warns it to trigger countermeasures, such as recovering to a safe state. The main reason for having such redundancy is that the monitor can be made substantially simpler than the main component, and more focused for the task at hand (e.g., anomaly detection).

Figure 1 illustrates a simple DNN-based monitoring architecture. First, the monitor is trained on nominal data to learn a model of normality. If the main system is also learning-based, it is advised to use the same training set used to train the main system. Then, by fitting a probability distribution of such data and selecting the accepted false alarm rate, we can determine a fixed initial threshold to be used in production [11].

The monitor is then used in the field to warn the main system if the inputs are regarded as unsupported. The monitor in Figure 1 is an instance of a *black-box* environment monitor, i.e., a monitor that analyzes the main system's input space (with no knowledge of its internal behaviour) and assigns an unexpectedness score, which should be low and below the threshold if such inputs are known/supported, or high and above the threshold otherwise.

AEs are often used as black-box anomaly detectors because they bring several advantages [10]. First, they are independent from the monitored architecture, requiring no modifications to the main system, because they use information which is readily available for analysis. Second, their training process is quite straightforward and efficient, even with large datasets. Third, AEs are able to learn meaningful latent spaces even from a few examples unsupervisedly. Last, they are extremely fast at making predictions once trained, which makes them suitable for runtime, in-field monitoring techniques [11].

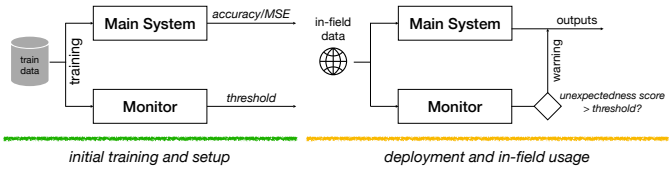

Fig. 1: Training and usage of a static DNN-based monitor

However, the main prerequisite of this architecture is that the main system and the monitor should have analogous generalization capabilities. When used in production, the observed data may differ from those used for training. There could be cases in which the main system generalizes better than the monitor, and then false alarms would be *mistakenly* reported. Perhaps worse, if the monitor generalizes too much to hazardous scenarios which are unsupported by the main system, true alarms could be *missed*. However, training two different DNNs to have comparable generalization capabilities is a challenging task. Thus, a more practical solution requires *adapting* the anomaly detection capabilities of the monitor as new knowledge becomes available.

III. Challenges of Adaptation

A. The Need for Adaptation

AEs are limited in efficiently incorporating *new* knowledge after they have been trained. Moreover, the thresholds used for anomaly detection are also determined offline before they operate in the field which is clearly suboptimal if the anomaly detector operates in a constantly changing environment. For nontrivial operational domains, it is virtually impossible to account for all possible nominal scenarios at training time. Thus, it is necessary to establish accurate and efficient online self-adaptation mechanisms, to integrate new types of knowledge, both related to novel nominal classes of data, as well as to anomalous ones.

In the context of DNNs, one way to tackle this problem is by using continual learning (CL), that is the ability of a machine learning model to continually learn from data. In the literature, CL is a multi-faceted term and use cases vary tremendously. For instance, in multi-task learning, a model learns to perform first a task A from a batch of data, and then a second task B from another batch of data that was not available during the training of task A. The model is then evaluated for its capability to correctly perform task B, without forgetting the earlier task A (*catastrophic forgetting*). Another application of CL deals with data that are discarded after training, due to privacy reasons (e.g., data of patients in a hospital) or unavailability (e.g., a weather forecasting systems processing a continuous stream of data).

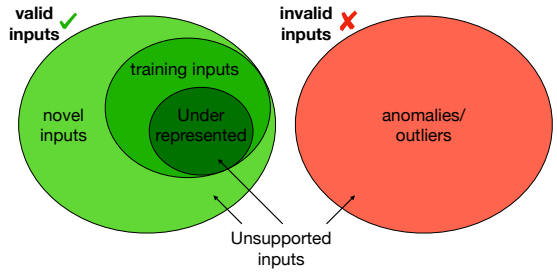

Fig. 2: Classification of unsupported inputs

In this paper, by CL we mean the ability of a model to autonomously learn and self-adapt in production as new data becomes available. For what concerns the adaptation of the anomaly detector, several challenges need to be addressed: (1) **effectiveness**, i.e., what data to consider for retraining. (2) **performance**, i.e., when it is appropriate to retrain the monitor. An adaptive anomaly detector should reactively adapt to dataset changes. However, short-term changes to the data distribution should not cause false alarms or frequent model updates, while permanent context drifts should trigger model updates as soon as possible. (3) **dependability**, i.e., how to safely update the monitor. The monitor would need to be updated as knowledge of anomalies and novel data accumulates, without however affecting its main functionality of providing a fail-safe redundancy mechanism (i.e., the update of the old monitor with the new one should not cause any service interruption). (4) **revalidation**, i.e., how to regression test the new monitor. The new monitor should continue to meet its key requirements, hence a regression testing mechanism should be implemented to mitigate catastrophic forgetting.

In the context of unsupervised anomaly detection, the main advantage of using CL is that *no labeling is required for the newly collected data*. In this work, we propose and evaluate metrics that can help automate the decision process of determining what data to consider for retraining.

B. Classes of Unsupported Inputs

A major drawback of DNNs consists in their inability to distinguish supported (valid) from unsupported (invalid) inputs. In fact, given a numeric input vector, a DNN will produce its output even if such input is meaningless in the domain where the DNN is supposed to operate [19]. Thus, anomaly detection techniques are used to avoid DNNs processing unsupported inputs, which may cause unpredictable predictions, potentially causing severe system-level failures, if not handled properly.

To gain an understanding of the types of unsupported input that can occur in real-world situations, Figure 2 categorizes them into three classes based on their proximity to the training set data distribution. The figure highlights the complexity and the variety of scenarios that CL monitors should aim at.

Supported inputs pertain to the same data distribution of the ones used for training (*in-distribution inputs*) and therefore they *should* be handled correctly by the DNN (actually, the proportion of correctly handled inputs—aka *accuracy*—is a statistical property of a DNN, which depends on the quality of the training data and of the training process, like data representativeness, balancing, diversity, and a careful model's architecture and hyper-parameter tuning).

Unsupported inputs, on the other hand, pertain to a different data distribution than the one used for training (*out-of-distribution inputs*). Underrepresented inputs is a first category of inputs that can cause a failure of the monitor in learning meaningful patterns for such data, due to their low frequency of occurrence, as compared to other, more represented, classes. A second category consists of *novel data*, i.e. data in the domain of validity of what the DNN *should* support, but which are not yet represented at all in the set used for training and should therefore be added to it as representative of new classes of data. This category is the main target for CL and adaptation tasks. Finally, the third category of unsupported inputs that can be found in real-world domains consists of *anomalous data*, i.e., inputs that are not in the validity domain and, as such, should be recognized and discarded.

IV. Continual Learning Anomaly Detection

The goal of our approach is to extend an existing monitor $\mathcal{M}$ that uses a static threshold for anomaly detection (see Figure 1) by equipping it with online learning ability, which encompasses incremental model updating strategies, with a focus on the unsupported input classes described in Section III-B. Our idea revolves around an interplay between the main system and the monitor. By establishing a perpetual information exchange cycle, we collect in-field performance indicators of the main system's behaviour to drive the improvement of the monitor.

Our framework leverages two opportunities. First, it uses in-field behavioural indicators to refine the definition of positive and negative cases. Second, it collects and relies on data that require no label as AEs are trained with no supervision.

More specifically, the detection of positives (i.e., samples whose reconstruction errors are above the threshold) and negatives (i.e., negative samples whose reconstruction errors are below the threshold) is based on a black-box analysis of the sole inputs and it is immaterial to how the system is actually behaving in the field in response to such inputs. In the absence of a ground truth, we hybridize the definition of positive and negative cases for anomaly detection by taking into account the main system's behaviour. The observed samples are regarded as empirically valid, since they have been collected during the nominal execution of the system, as far as the feedback from the system is correct (i.e., no safety oracle is violated).

It is important to highlight that our focus is on updating and improving the monitor, and not the main system. If, during the CL phase, there is evidence that the faced data distribution shift is too large to be managed correctly by the main system (i.e., safety oracles are repeatedly violated), then in-field monitoring is no longer an option and the main system has to be shut down for additional offline retraining and testing. The collected data can be used for such retraining (at the price of manual labeling operations), or to produce test cases that mimic the novel conditions found in the field [19].

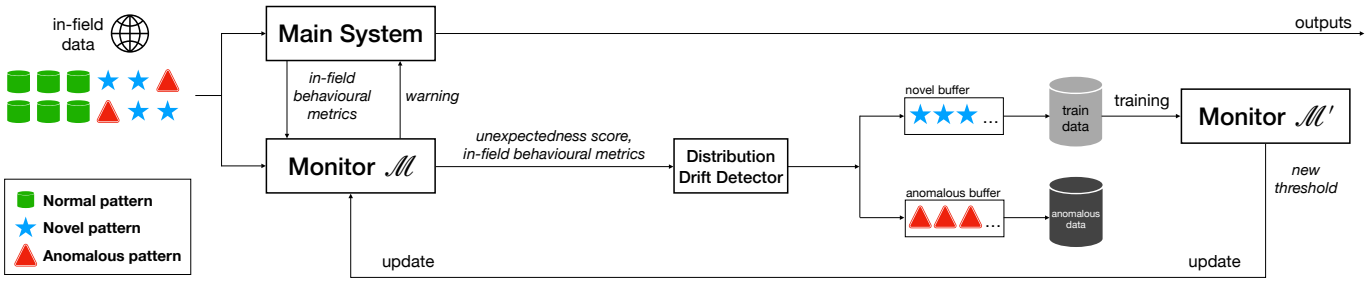

Fig. 3: Our CL monitoring framework for anomaly and novelty detection with in-field behavioural metric guidance

A. Approach

Figure 3 illustrates the usage scenario of our framework. In a nutshell, the monitor not only predicts the unexpectedness scores, but it also receives runtime behaviour indicators data as the main system executes. Once the monitor is initialized, it is ready for online prediction. The main thread works on real-time anomaly detection, whereas a secondary thread (Distribution Drift Detector) collects behavioural in-field data from the main system continuously.

Distribution Drift Detector. The distribution drift detector determines what data to store for adaptation. It uses in-field behavioural metrics to understand whether the samples unsupported by $\mathcal{M}$ are also unsupported by the main system.

Upon the occurrence of alarms raised by the monitor or when behavioural metrics indicate poor system performance, anomalous data are stored into an anomaly buffer for later evaluation. When no alarm is raised and behavioural metrics indicate acceptable system performance, but data are classified as novel, the collected data are added to the training set of the monitor $\mathcal{M}$, which is ready for retraining.

Triggering Adaptation and Model Update. During the online processing, if the monitor detects that the model no longer fits the current data, then model updating is triggered. At least two options are possible. A first option consists in adding the newly collected data to the training set and retraining the monitor on the whole resulting set. Another option is incremental training. After retraining, a new threshold is also computed, by estimating the shape and scale parameters of the fitted Gamma distribution of the reconstruction errors and selecting the desired accepted false alarm rate [11]. Finally, the new monitor $\mathcal{M}'$ seamlessly substitutes the monitor $\mathcal{M}$.

Several strategies are possible to decide when to trigger model updating, such as setting a limit to the buffer's size, or a time limit, possibly combined with the former. Our general guideline is that small distribution drifts can be rapidly incorporated, hence a small buffer size would help achieve fast adaptation. However, a drawback is that the retraining operation is quite computationally and time demanding for an online setting, thus if the buffer size is too small, there is a chance that the monitor will be consuming too many computational resources due to frequent retraining operations. Selecting universal optimal values for such parameters, or an efficient retraining strategy, is beyond the scope of this paper.

V. Case Study and Preliminary Evaluation

We performed a preliminary study to assess the effectiveness of our framework to improve an anomaly detector from the literature. Effectiveness is defined as the capability of the anomaly detector at maintaining a high detection rate while keeping the false alarms low in presence of distribution shifts.

A. Case Study

We exemplify our framework on a self-driving car case study: the Udacity simulator from the work by Stocco et al. [11]. The simulator consists of different closed-loop track circuits and supports training and testing of an autonomous driving system that performs *behavioural cloning*, i.e., the DNN autopilot learns the *lane keeping* functionality from a dataset of driving scenes collected from a human driver. The simulator is equipped with features to inject controllable operational conditions, such as weather changes, thus it can be used to produce different test datasets having both supported and unsupported inputs. For instance, in the work by Stocco et al. [11], the nominal condition was set to sunny weather, whereas the unsupported conditions were set to either rain, fog, snow, or day/night shift. The simulator was used to test the effectiveness of different AEs as black-box runtime misbehaviour predictors of the self-driving car confidence. In particular, the variational autoencoder (VAE) yielded the best detection results. However, the reported false alarms rate in nominal conditions is non negligible. This suggests that there is potential for adaptation by continually learning the underrepresented inputs from the field. In this paper, we use the trained DNN-based self-driving car model called DAVE-2 [2] and we investigate whether our approach can improve the trained VAE anomaly detector.

B. In-field Behavioural Metrics

As required by our framework, we need an automated way to determine internal and behavioural metrics for the main system. We implemented two in-field metrics concerning the quality of driving, namely *predictive uncertainty* and *lateral deviation*. Table I shows how we use such internal and behavioural metrics (respectively, *Unc* and *CTE*) to classify the collected data as likely true/false positive/negative cases (LTP, LFP, LTN, LFN), along with the thresholds used in our study. Adaptation of the anomaly detector makes use of the

TABLE I: Classification of likely true/false positive and negative cases (LTP, LFP, LTN, LFN) for anomaly detection adaptation based on either context, uncertainty, or lateral deviation

Metric • Threshold	Rule
Autoencoder Loss [11] • $t = 0.0472$	
	LTP $\rightarrow context = unexpected \wedge loss > t$
	LTN $\rightarrow context = nominal \wedge loss < t$
	LFP $\rightarrow context = nominal \wedge loss > t$
	LFN $\rightarrow context = unexpected \wedge loss < t$
MC-Dropout • $t_1 = 0.00328$ [20]	
	LTP $\rightarrow Unc > t_1 \wedge loss > t$
	LTN $\rightarrow Unc < t_1 \wedge loss < t$
	LFP $\rightarrow Unc < t_1 \wedge loss > t$
	LFN $\rightarrow Unc > t_1 \wedge loss < t$
CTE • $t_2 = 1.5$ (mt)*	
	LTP $\rightarrow CTE > t_2 \wedge loss > t$
	LTN $\rightarrow CTE < t_2 \wedge loss < t$
	LFP $\rightarrow CTE < t_2 \wedge loss > t$
	LFN $\rightarrow CTE > t_2 \wedge loss < t$

* *considering the width of the vehicle in the Udacity simulator.*

LFP data, i.e., those underrepresented or missing inputs that cause the anomaly detector to trigger a false alarm.

Predictive Uncertainty. The first considered metric is a white-box internal measure of the system's confidence in its own predictions. We use the predictive variance of dropout-based DNNs called Monte Carlo (MC) dropout, which we will refer hereafter simply to as MC-Dropout [21]. Essentially, by collecting multiple predictions for a single input, each with a different realization of weights due to dropout layers (i.e., regularization layers which help prevent overfitting), it is possible to account for model uncertainty in a DNN at testing time. For a complete overview of MC-Dropout, we refer the reader to the relevant literature [21].

MC-Dropout has been proposed in the self-driving car domain [20] as a measure to approximate uncertainty information from DNNs that perform regression problems, such as DAVE-2. For the implementation of MC-Dropout-based DAVE-2 model we followed the guidelines provided in a similar experiment [20].

The rationale for using MC-Dropout is that supported inputs are expected to be characterized by low DNN uncertainties, whereas unsupported inputs are expected to increase it.

Lateral Deviation. In our setting, the car is trained to follow the center of the road. Thus, the second considered metric is a black-box measure of the car's distance from the center of the road, which we refer to as *lateral deviation*. The tracks in the Udacity simulator are equipped with waypoints, i.e., phantom objects that are used to mark distinct sectors. We implemented a component within the Udacity simulator that measures the cross track error (CTE), which is the distance from the center of the car's cruising position to the center of the road on the ideal trajectory between the planned route given by two consecutive waypoints. The rationale for using lateral deviation is that unsupported inputs may cause erroneous steering angle predictions, thus lowering the chances of the self-driving car to follow the ideal trajectory.

C. Procedure

For this initial evaluation of our framework, we performed two studies, concerning *class imbalance* (i.e., underrepresented inputs), and *novelty detection* (i.e., novel inputs), respectively. To this aim, we executed several two-lap simulations on the first circuit provided by the Udacity simulator (Lake Track).

For evaluating *class imbalance*, we performed one simulation in the same nominal conditions as the training set, using the VAE anomaly detector. This allowed us to precisely collect the number of false alarms (false positives) in nominal conditions, as well as the in-field behavioural metrics MC-Dropout and CTE. Then, we used our framework to detect the occurrences of underrepresented samples to be used for adaptation, i.e., the likely false positives (LFP) according to the ruleset in Table I.

For evaluating *novelty detection*, we performed one simulation activating a single unexpected condition, namely rain. We kept the rain particles emission rate low (i.e., between 100 and 1,000 particles per second, which represent respectively $1/100$ and $1/10$ of the settings used by Stocco et al. [11] for the same condition) so as to execute the car in a slightly novel condition that would not be however too extreme, hence causing too many system-level failures (i.e., true positives: crashes or car going out of track).

As suggested for a similar experiment [20], for the MC-Dropout predictions we used a batch size of 128 and the car's maximum speed was reduced to 15 mph (it was 30 mph during data generation), which provides a good trade-off between processing time and accuracy.

After collecting new samples for the likely false positives, we retrained the VAE offline. We added copies of instances from the underrepresented class by oversampling them (i.e., by sampling with replacement multiple times). Finally, we executed further simulations to determine the number of false positives after adaptation of the anomaly detector.

To assess *effectiveness*, we measured the number of false positives detected in each configuration of our framework, using the threshold prior to the adaptation phase.

D. Results

Table II presents the effectiveness results for the two studies. The table shows the number of likely false positives detected by MC-Dropout or CTE *before* and *after* adaptation, both numerically (Columns 2 and 6) and percentage-wise (Columns 3 and 7). The other columns show the false positive rate when the confidence threshold is set to $\epsilon = 0.05$ (i.e., at TPR = 95%), along with the false negative rate.

In the class imbalance study, we experienced 64 likely false alarms. Our framework detected nearly all of them (98%) when equipped with MC-Dropout, and about half of them (46%) when using CTE. This essentially means that, for such frames, the self-driving car was driving quite confidently. The results after adaptation (VAE retraining) confirm our hypothesis, as almost no likely false positives are observed. It is quite important to highlight that TPR/FNR values are not reported for this experiment as no system-level failures

TABLE II: Evaluation results: likely false positives and false positive / true negative rates before and after adaptation, by distribution drift detector (either MC-Dropout or CTE)

CLASS IMBALANCE	*in-field collection*				*after adaptation*			
	LFP (#)	**detect rate (%)**	**FPR at TPR 95%**	**TNR**	**LFP (#)**	**detect rate (%)**	**FPR at TPR 95%**	**TNR**
VAE	65	-	5.12	94.88	1	-	0.09	99.91
VAE & MC-Dropout	64	98	0.08	99.92	1	100	0.09	99.91
VAE & CTE	30	46	2.76	97.24	1	100	0.09	99.91
NOVELTY DETECTION	*in-field collection*				*after adaptation*			
	LFP (#)	**detect rate (%)**	**FPR at TPR 95%**	**TNR**	**LFP (#)**	**detect rate (%)**	**FPR at TPR 95%**	**TNR**
VAE	143	-	13.48	86.52	13	-	1.05	98.95
VAE & MC-Dropout	134	94	0.85	99.15	12	92	0.08	99.92
VAE & CTE	65	45	7.35	92.65	11	85	0.16	99.84

occurred when driving in nominal conditions (this means that all likely false alarms are false alarms, in this experiment).

In the novelty detection study, we experienced more likely false alarms (143), which was expected, as the car was driving in relatively novel conditions. Also in this case, our framework detected a large number of likely false positives (94%) when equipped with MC-Dropout. The detection rate of CTE (45%) is inline with the previous study. Results after retraining also confirm that most likely false alarms are eliminated thanks to adaptation. Two system-level failures occurred when driving in the novel conditions of the second study. Both were correctly detected (i.e, TPR=1 and FNR=0; these results are not reported in the table for conciseness and presentational clarity).

VI. DISCUSSION

Analysis of the Results. Our preliminary evaluation revealed that the VAE anomaly detector is not able to learn meaningful nominal patterns if class imbalance affects the training set. To this aim, our CL framework provides promising results concerning minimization of false alarms, improving an existing state-of-the-art autoencoder-based anomaly detector within a real-world context. In this work we consider black-box environment monitors, specifically VAEs, even though our adaptation framework is quite generic, and the notions and ideas proposed in this paper are applicable to other kinds of monitors and architectures.

We found that the DAVE-2 model generalizes better (or differently) than the VAE-based anomaly detector in case of class imbalance. Our results show that it is possible to use MC-Dropout to obtain reliable uncertainty scores, confirming previous studies [20]. This might suggest that it would be possible to build an anomaly detector relying solely on white-box metrics. However, such results should be taken with care as there are significant drawbacks that must be addressed before they can be used in production. Despite the accuracy of the MC-Dropout predictions, we observed that it is indeed quite computationally demanding for an online setting. Even if we do not report extensive performance results, in summary we have observed that the performance of the main system and the overall driving quality were affected to a non negligible extent when increasing the speed of the car (> 15 mph and ≤ 30 mph) or the batch size. Thus, MC-Dropout might not be the best choice for online runtime drift detection, unless accelerator-specific hardware is available within the deployed system (e.g., tensor processing units).

Even if CTE is commonly used in other domains, such as aircraft landing systems [22], in the self-driving car domain it is not a standard metric as it may not always be available, since it requires the capability to compare the ideal with the actual trajectory. Current industrial self-driving cars can accurately approximate metrics such as CTE by means of advanced computer vision algorithms. In the future, CTE may become more standard and utilized, perhaps with the advent of smart cities and dedicated infrastructures for self-driving vehicles, which could provide online information about the car's driven lane position.

Our studies confirmed that VAE alone experiences many false alarms on novel conditions. The best configuration from our experiments is to use the VAE anomaly detector, paired with MC-Dropout as drift detector. This allowed us to remove most false positives, mistakenly reported when the car was driving safely. With around half drift detection rate, CTE is suboptimal as a drift detector in comparison to MC-Dropout. Still, it contributed to a competitive reduction of the likely false positive rate. Hence, CTE might be a good trade-off between drift detection rate and false positive reduction, especially in resource constrained settings, where the available hardware is not compatible with the expensive online uncertainty computation performed by MC-Dropout.

Applications. The data mined from the field can be also used for retraining of the main system. However, such data lack labels, which in practice are costly to get as labeling requires human effort. In the case of a self-driving car, this problem is even more difficult. Indeed, it is quite challenging if not impossible to manually assign a meaningful label (i.e., a steering angle) only by looking at individual images. A more accurate but expensive option would require a human driver to drive in a simulated environment as close as possible to the parts of the track in which the likely false positives were detected. The control actions (e.g., steering angles) performed by the human would then serve as ground truth labels. Au-

tomated test generation techniques could be used to produce a simulation environment that mimics the underrepresented conditions found in the field. Another, cheaper option could be to search the training set for samples that are close to the ones collected in the field (e.g., the k-nearest neighbours based on some image similarity metric such as the SSIM [23]). Data augmentation could be used to artificially add slightly modified instances of such conditions to the training set.

Another interesting direction would be to exploit the knowledge of true anomalies (i.e., true system failures) for retraining a better monitor. With the current setup (i.e., autoencoders) negative examples cannot be used, as the AE can only learn patterns of nominal data, not anomalous ones. However, researchers have proposed more sophisticated AE architectures in which, as new knowledge of anomalies become available, such information can be leveraged to build better anomaly detectors. For instance, one way to do so could be by combining AEs with a triplet loss function [24]. Triplet loss is used to minimize the distance (maximize the similarity) between in-distribution and positive samples, while maximizing the distance (minimizing the similarity) between in-distribution and negative samples.

Open Challenges. One major challenge is how to deploy new models obtained after adaptation of the anomaly detector without negatively affecting the main system's behaviour. Moreover, it is important to strike a balance between old and new knowledge by maintaining also some samples of older data distribution within an archive, to avoid overfitting towards only the most fresh data that is collected during runtime adaptation. Techniques should be devised to maintain the training set size limited only to the most representative examples, as the retraining time scales up with the size of the training set. Training set minimization techniques could be used, for instance based on more sophisticated sampling techniques, such as stratified sampling or Synthetic Minority Oversampling Technique (SMOTE) [25].

Another open challenge concerns the coupling between the main system and the monitor. Relying on in-field behavioural metrics may diminish the anomaly detector's effectiveness in cases whereby the main system produces reasonable in-field behaviour also for inputs that should be regarded as anomalies. Thus, including such inputs in the anomaly detector's training set might introduce a drift in the monitor's knowledge, which may become less sensitive to behaviours similar to the undetected anomaly. In our experiments, this situation did not occur, as the two system-level failures were correctly detected by the monitor both pre and post adaptation. However, the coupling between the main system and the safety monitor may have far-reaching consequences that require further investigation.

VII. Related work

Online/Adaptive Anomaly Detection. Yuan et al. [26] present an online anomaly crowd detection method for pedestrian detection, using a context model called structural context descriptor (SCD). Anomalies are detected by temporal and spatial analysis of the SCD variation over time. Sun et al. [27] study how to adaptively adjust the detection threshold of Intrusion Detection Systems (IDSs) in the context of cellular mobile networks. In particular, they use Shannon's entropy measure to identify the uncertainty of the up-to-date normal profile. Cretu-Ciocarlie et al. [12] propose to improve anomaly detection of intrusion detection systems by enhancing the training of anomaly detector sensors with a self-calibration phase, leading to the automatic computation of the optimal parameters. Liao et al. [28] present an adaptive anomaly detection framework based on the use of unsupervised evolving connectionist systems. Huang et al. [29] adapts an anomaly detector based on Recurrent Neural Network (RNN) with a Reinforcement Learning (RL) method to achieve the self-learning process.

While our approach falls in the category of unsupervised anomaly detection, in contrast to the existing works, we study specifically how to adapt an AE-based anomaly detector in the self-driving car domain, using in-field quality metrics (predictive uncertainty and lateral position) as drift detectors. Our architecture and its instantiation for the self-driving car domain, represents a unique and novel contribution to the state of the art.

Online/Adaptive Autoencoders. Kim et al. [30] aim at fine-tuning the performance of an already-trained denoising autoencoder (DAE) in the context of semi-supervised audio source separation. Doshi et al. [31] propose an online anomaly detection method for surveillance videos using transfer learning and continual learning. Wiewel et al. [32] study how to mitigate catastrophic forgetting within VAEs when trained on a continually growing set of nominal data. In another paper, Wiewel et al. [33] target novel class detection with one-class classification.

To the best of our knowledge, our study is the first that combines CL with in-field metrics, such as predictive uncertainty, to detect distribution drifts of the input data and to drive the retraining of a better VAE.

VIII. Conclusions and future work

This paper proposes a framework for continual learning of a runtime monitoring system, which keeps evolving an anomaly detector as additional experience of nominal and anomalous instances become available. When the observed instances deviate from the nominal distribution of the data used for training the anomaly detector, new samples are collected and used for adaptive retraining. Our experimental results show that uncertainty and behavioural metrics can be used as drift detectors, and that the resulting reduction of the false alarm rate is substantial.

In our future work we plan to address the remaining challenges, which include the safe deployment of adapted monitors, the trade-off between frequent adaptations and introduction of regressions, and the exploitation of knowledge about true anomalies observed in the field.

ACKNOWLEDGMENTS

This work was partially supported by the H2020 project PRECRIME, funded under the ERC Advanced Grant 2017 Program (ERC Grant Agreement n. 787703).

REFERENCES

[1] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing Machine Learning based Systems: A Systematic Mapping," *Empirical Software Engineering*, 2020.

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, vol. abs/1604.07316, 2016.

[3] Q. Zhang, H. Wang, H. Lu, D. Won, and S. W. Yoon, "Medical image synthesis with generative adversarial networks for tissue recognition," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2018.

[4] J. Zhao, Q. Feng, P. Wu, R. A. Lupu, R. A. Wilke, Q. S. Wells, J. C. Denny, and W.-Q. Wei, "Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction," *Scientific Reports*, vol. 9, no. 1, p. 717, 2019. [Online]. Available: https://doi.org/10.1038/s41598-018-36745-x

[5] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *CoRR*, vol. abs/1810.04240, 2018.

[6] I. Žliobaitė, M. Pechenizkiy, and J. Gama, *An Overview of Concept Drift Applications*. Cham: Springer, 2016, pp. 91–114.

[7] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320318302565

[8] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.

[9] W. Wu, H. Xu, S. Zhong, M. R. Lyu, and I. King, "Deep validation: Toward detecting real-world corner cases for deep neural networks," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 125–137.

[10] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009.

[11] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of 42nd International Conference on Software Engineering*, ser. ICSE '20. ACM, 2020.

[12] G. F. Cretu-Ciocarlie, A. Stavrou, M. E. Locasto, and S. J. Stolfo, "Adaptive anomaly detection via self-calibration and dynamic updating," in *Recent Advances in Intrusion Detection*. Springer, 2009.

[13] U. Fiore, A. D. Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, 2019.

[14] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "Iot healthcare analytics: The importance of anomaly detection," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016.

[15] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *CoRR*, vol. abs/1901.03407, 2019.

[16] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12, (NIPS)*, 1999.

[17] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*, 4th ed. Wiley Publishing, 2009.

[18] T. Kohonen, *Self-Organizing Maps, Third Edition*, ser. Springer Series in Information Sciences. Springer, 2001.

[19] V. Riccio and P. Tonella, "Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing," in *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE '20, 2020.

[20] R. Michelmore, M. Kwiatkowska, and Y. Gal, "Evaluating uncertainty quantification in end-to-end autonomous driving control," *CoRR*, vol. abs/1811.06817, 2018.

[21] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML '16. JMLR.org, 2016.

[22] D. D. M. Tyler C. Staudinger, Zachary D. Jorgensen, "XPlane-ML - an Environment for Learning and Decision Systems for Airplane Operations," in *Proceedings of Machine Learning Open Source Software 2018*, ser. MLOSS '18, 2018.

[23] R. Yandrapally, A. Stocco, and A. Mesbah, "Near-duplicate detection in web app model inference," in *Proceedings of 42nd International Conference on Software Engineering*, ser. ICSE '20. ACM, 2020.

[24] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, Jun. 2002.

[26] Y. Yuan, J. Fang, and Q. Wang, "Online anomaly detection in crowd scenes via structure analysis," *IEEE Transactions on Cybernetics*, vol. 45, 2015.

[27] Bo Sun, Zhi Chen, Ruhai Wang, Fei Yu, and V. C. M. Leung, "Towards adaptive anomaly detection in cellular mobile networks," in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 2, 2006.

[28] Y. Liao, V. R. Vemuri, and A. Pasos, "Adaptive anomaly detection with evolving connectionist systems," *Journal of Network and Computer Applications*, vol. 30, no. 1, 2007, network and Information Security: A Computational Intelligence Approach.

[29] C. Huang, Y. Wu, Y. Zuo, K. Pei, and G. Min, "Towards experienced anomaly detector through reinforcement learning," in *AAAI*, 2018.

[30] M. Kim and P. Smaragdis, "Adaptive denoising autoencoders: A fine-tuning scheme to learn from test mixtures," in *Latent Variable Analysis and Signal Separation*. Springer International Publishing, 2015.

[31] K. Doshi and Y. Yilmaz, "Continual learning for anomaly detection in surveillance videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2020.

[32] F. Wiewel and B. Yang, "Continual learning for anomaly detection with variational autoencoder," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[33] F. Wiewel, A. Brendle, and B. Yang, "Continual learning through one-class classification using vae," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.