Anomaly Detection, Active Learning, Precursor Identification, and Human Knowledge for Autonomous System Safety

Nikunj C. Oza,* Kevin M. Bradner,† David L. Iverson,‡ Adwait Sahasrabhojanee,§ and Shawn R. Wolfe¶
Intelligent Systems Division, NASA Ames Research Center, Mail Stop 269-3, Moffett Field, CA 94035-0001, USA

The project Autonomy Teaming and TRajectories for Complex Trusted Operational Reliability (ATTRACTOR) [1] researched and developed Artificial Intelligence with application to multi-Unmanned Aerial Systems (UAS) missions. Such missions, like other complex systems-of-systems, are likely to have previously-unknown, safety relevant anomalies occur due to many possible factors including system failures or degradations, emergent behavior, changes in the environment in which the systems operate, changes in the way the systems are operated. We discuss the application of anomaly detection, active learning, and precursor identification to identify such anomalies and the conditions under which they are more likely to appear. We demonstrate results on simulated multi-UAS missions that show promise to be applied to real missions.

I. Introduction

In this work, we are interested in finding examples of operations that are anomalous because of conditions under which there is a higher than normal probability of mission failure. The airline industry and others involved in commercial aviation primarily look for known examples of anomalous operations using exceedances. These are rules that specify thresholds over several variables which, if violated, trigger the exceedances. Often a severity with which the threshold is violated is also included. However, by definition, these cannot find previously unknown anomalies. Additionally, developing and refining these exceedance-based systems requires significant domain expert time.

We have developed several machine learning-based techniques for anomaly detection which essentially allow the data to "speak for themselves." These techniques learn, from past data, a model that represents normal behavior, and then use that model in the future to determine, for new data, whether they represent normal or anomalous operations. These methods require significantly less domain expert time to develop than exceedance-based methods—the time that domain experts have spent operating the system in the past can be leveraged by using the resulting data to train the anomaly detection algorithm. Using these data represents an indirect way that humans provide input to the algorithm, which should increase trust in the algorithm and the anomalies that it returns.

Given how safe the aviation industry is, we can safely assume that data that are in high probability density regions, which the anomaly detection algorithm identifies as normal, actually represent normal and safe operations. However, data identified as statistically anomalous may not actually represent abnormal or unsafe operations—for example, they may be data errors or recovery actions that look unusual. These cases represent false alarms. While false alarms are better than missed detections, too many false alarms leads to alarms being ignored. For this reason, we have developed a process that uses active learning to obtain a label from a subject matter expert (SME) on whether a statistical anomaly is operationally significant, and the rationale for his/her decision. Active learning is the area of machine learning that assumes that there is a cost to labeling data and so tries to minimize the number of examples that require labeling. Therefore, active learning tries to find examples that, if labeled by the SME, would maximally improve the desired metric, such as, in our case, false alarm rate. In addition to improving the false alarm rate, our process has the benefit of involving SMEs in the process of developing and refining the anomaly detection system, thereby increasing trust in the system. SMEs can continue providing such feedback after the anomaly detection system is deployed as they have time and as needed to improve the system further and reflect changes in normal and anomalous behavior over time.

In addition to finding anomalies themselves, we are interested in finding the situations that increase the probabilities of these anomalies occurring in the future. We have developed precursor identification algorithms that, given normal

*Leader—Data Sciences Group
†Computer Scientist—Data Sciences Group
‡Computer Scientist—Data Sciences Group
§Computer Scientist—Data Sciences Group
¶Computer Scientist—Ground and Flight Data Systems Group

data and data with anomalies, finds the precursors that, while possibly representing safe operations themselves, lead to an increased probability of the anomalies occurring. Domain experts can then determine whether the precursors should be mitigated or if the link between the precursors and anomalies can be broken through some change in operations or procedures.

In this work, we will demonstrate anomaly detection and active learning to reliably identify anomalous operations in simulated multi-UAS operations created as part of the project Autonomy Teaming and TRajectories for Complex Trusted Operational Reliability (ATTRACTOR) [1]. We will also demonstrate the identification of precursors to certain anomalies. The types of anomalies that we will investigate over the course of this study are both single-vehicle anomalies and multi-vehicle anomalies—in the latter, each vehicle may seemingly behave normally but the relationships between the vehicles (e.g., their coordination) may be anomalous.

As mentioned earlier, the current state of the art in the aviation industry is to use exceedances to identify known anomalies. Anomaly detection algorithms were developed before and after we developed the algorithms used in this report, although our algorithms were novel at the time that we published them. Anomaly detection algorithms have been used in other industries. However, we are the first, to our knowledge, to develop active learning as a follow-on to anomaly detection to reduce the false alarm rate. Active learning has, so far, only been used in the context of classification. Additionally, our precursor identification framework and the algorithms that we developed are novel.

II. Description of the Problem

The problem that we are working to solve is automatically detecting anomalies indicative of degraded system safety and performance, as well as the precursors to these anomalies.

The first problem that we address is identifying deviations from normal system performance. The problem here is to use data that represent normal operations and perhaps data that represent anomalous operations, and learn a model that can be used to label new data as representing normal or anomalous operations. There are two general ways that this problem can be solved. One method is to learn a probability distribution from past data. For new data, the probability distribution can be used to calculate the probability that the data could have been generated from that distribution. If the probability is lower than some threshold, then the data are assumed to represent anomalous operations, else normal operations. A second general method and the one that we used to solve this problem is to calculate, for new data, the distance to one or more nearest neighbors in the training data. The higher this distance, the more likely the data represent anomalous operations. Many measures or functions of distance to the nearest neighbors can be used. Often, instead of using the training data directly, summaries of the training data are used (e.g., clusters of training data). We use the Inductive Monitoring System (IMS) [2] and the Meta Monitoring System (MMS) for this problem. These algorithms cluster the training data and then evaluate new data by determining whether they fall in one of the clusters, in which case they represent normal operations, or else how far they are from the nearest cluster, in which case they represent anomalous operations, and the distances represent how severe the anomalies are. More details on the problem formulation and solution that IMS and MMS provide are given in sections V.A and V.B.

The second problem that we address is to distinguish, within the data representing anomalous operations, data representing operationally significant (OS) anomalies from data representing non-operationally significant (NOS) anomalies. We need domain experts to indicate whether these anomalies are OS or NOS, but we assume that domain experts have limited time to do this. Active learning is the branch of machine learning that attempts to minimize the amount of labeled data required by judiciously finding examples for which labeling will provide the most significant improvement. For example, within the context of anomaly detection described in the previous paragraph, one can ask domain experts to label examples that have an IMS cluster distance that is half of the maximum distance among all data points, since these are points halfway between normal and maximally anomalous. Our active learning framework is described in more detail in [3, 4] and in section V.C.

The third problem that we address in this work is to identify precursors to different anomalies. In particular, for one anomaly or a related set of anomalies, we want to find the earliest point in time at which an event occurred that increases the probability that the anomaly occurs in the future. For this problem, we assume that we have training data from full flights (in this case, each drone's flight separately) that are classified based on whether they contain an anomaly or not. Our algorithm, Deep Temporal Multiple Instance Learning (DT-MIL), uses the MIL framework, which assumes that if the flight is labeled as anomalous, then at least one data point within the flight is anomalous, while if the flight is labeled as normal, then all data points within the flight represent normal operation. Our algorithm uses this assumption to find parts of the flight that are consistent with anomalous operation, whether those parts are themselves anomalous or not. Additional details are provided in [5] and in section V.D.

III. Approach

When operating a complex engineering system involving multiple interdependently functioning components, the variety of ways in which problems can arise is often staggering. In some cases, these systems are given enough autonomy to potentially result in damaged equipment, human injury, or other unacceptable outcomes. To enable the monitoring of large quantities of data produced by these systems, it is often necessary to employ algorithms that can detect and report error states, or anomalies, as they arise.

Any method that claims to detect anomalies and their precursors for a wide variety of engineering systems must use general purpose mathematical techniques, as opposed to a solution specifically devised for the data being analyzed. In this work, we present such a technique based on several algorithms built upon principles of data science and probability. Data-driven methods have the advantage of being able to develop models that are grounded in real experimental observation of the system in question. Data science techniques perform well when matched with problems where a sufficient quantity of relevant data is available for a system, and that same system is too complex to cost-effectively create a manually defined model with comparable accuracy. The search-and-almost-rescue mission that has been the focus of ATTRACTOR satisfies these criteria.

In terms of data availability, a system with multiple remotely-commanded assets tends to provide a rich source of data. In a team of flying drones, each drone is likely to have onboard sensors reporting data such as position, velocity, heading, power supplied to each rotor, altitude, etc. In addition, an intelligent algorithm controlling one or more of these drones would likely be able to report the desired or expected physical state that it aims to put each drone into. Once this large supply of data is analyzed, anomalies that might not have been expected can nonetheless be identified. In addition to being simply hard to foresee, some anomalies that are discoverable from data analysis may be too subtle to detect via manual inspection. The following is an example of anomaly detection in a mission context involving the IMS algorithm, which we describe in Section V.

To demonstrate the utility of anomaly detection in a mission control setting, we built an IMS model to monitor temperature sensors in the wings of a Space Shuttle Orbiter, and used that model to analyze archived telemetry data collected from the ill-fated STS-107 Columbia Space Shuttle mission. This flight came to a disastrous end when the Columbia orbiter was destroyed during reentry, claiming the lives of all seven crew members. The ultimate cause of the accident was determined to be a breach in the Thermal Protection System on the leading edge of the left wing, caused by a piece of insulating foam that struck the wing approximately 82 seconds after launch. The first indication of the damage that was noticed by mission controllers monitoring telemetry data was not seen until Orbiter re-entry, 17 days after launch. That anomaly was a slight increase in a brake line temperature of the left main landing gear that occurred about seven minutes before the destruction of the vehicle.

IMS models for the launch and ascent analysis were generated from training data collected during 10 previous Columbia flights. Separate models were generated for each wing. Training vectors were formed from four corresponding temperature sensors in each wing of the Orbiter. The results of the IMS analysis of the STS-107 Columbia launch are graphed in Figure 1. The horizontal axis represents time, beginning at the moment of lift off. The vertical axis represents the IMS measure of deviation from nominal behavior. IMS monitoring results for the left wing are represented by the lighter colored line. Right wing results using corresponding sensors are shown as a darker line for comparison. The vertical line near time 15:40:22 shows the moment of the foam impact event that breached the Thermal Protection System on the left wing. Notice that the IMS results for the left and right wings remain fairly low until shortly after the foam impact, at which point the values for the left wing begin to diverge sharply. The IMS deviation values for the right wing continue to show results within a reasonable range of nominal, while the left wing deviation values increase to nearly three times those of the right wing.

Although this STS-107 analysis was performed off-line using archived data, the techniques used could be implemented for real time monitoring. Significant deviations in a group of sensors or asymmetrical results for identical sensor sets, especially of large magnitude such as shown in this analysis, may provide indication of anomalies earlier in a mission than current monitoring tools.

As with any method based in data science, the disadvantage of this approach compared to a manually crafted anomaly detection model is its reliance on high quality data and, for some methods, a large volume of such data. It is worth noting what is meant by both *large volume* and *high quality*. For more information, see Section IV.

The primary data source used in this work is a search and rescue inspired drone simulation. The simulation was implemented in the Unity game engine, and it provides for a team of drones flying a precomputed pattern (Figure 2). It also includes a few mechanisms for introducing anomalies. The first of these methods involves causing a rotor failure to occur on a particular drone. The other main way of introducing anomalies involves trees in the virtual environment. Drones may run into these trees as a result of a failure in the collision avoidance mechanism. In addition, other sources

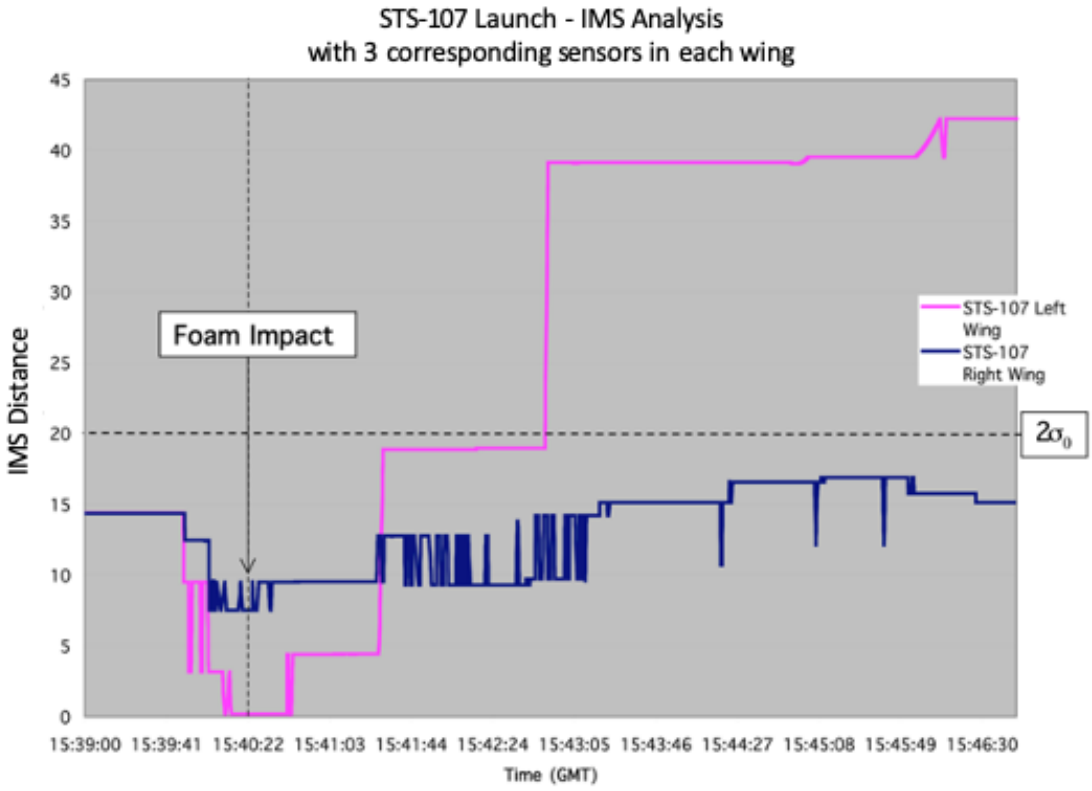

Fig. 1 Results of IMS analysis of STS-107 Columbia launch

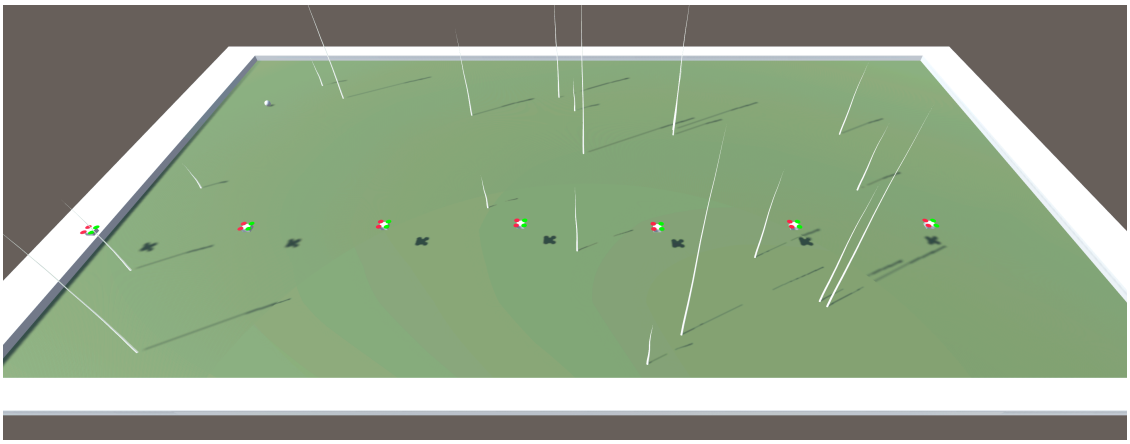

Fig. 2 Simulation with seven drones and twenty trees.

of collisions and anomalies are possible by setting parameters of the simulation that were not specifically designed to introduce anomalies. For more information, see Section VI.

IV. Assumptions and Dependencies

The foremost dependency of any data science application is relevant data. Under many circumstances, the decision to undertake a data science project is made in response to the availability and apparent potential of existing data. Because the ATTRACTOR project saw development of data science models concurrently with the development of the data they rely on, substitute data was generated by the data science team. The quality of these data was necessarily limited by a desire to spend more development time on the models that would eventually deal with the real data supplied by other efforts within the ATTRACTOR project. The resulting data generation software, a Unity simulation of drones performing a search-and-almost-rescue operation, was therefore optimized for suitability as an input to a data science algorithm. Of course, more realistic sources of data often see their designs dominated by other considerations, resulting in less than ideal opportunities to learn from the data they produce. The following considerations can be used to design a system suitable for anomaly detection using our approach.

In the case of anomaly detection, it is important that the data provided include enough information to characterize a wide variety of system activity. Ideally, this variety of states would be fully representative of all nominal system states. In many applications of practical interest, this corresponds to tens of thousands of data entries which correspond to hundreds of ordinary system operations. The quality of data is somewhat harder to define. At the least, data must be reported accurately and be mostly complete (i.e. few missing sensor values) in order to be considered high quality. Beyond this, though, it is important that data supply the right kind of information. For instance, a learned model would be able to predict crashes between a pair of robots more reliably when these robots publish their positions directly than when they report velocity or accelerometer data. Regardless of whether the eventual data science techniques will rely directly on sensor data or on quantities derived from it, it is important to make sure that these data are reported accurately. In many cases, the best way to ensure this is by using sensors of a high quality.

In addition to data from nominal system operation, it is often useful to supply data describing the system in an anomalous state. Some of our anomaly detection techniques can use these data for training. Even for our algorithms that do not train with such data, they nonetheless serve as a useful means of evaluating and quantifying the performance of a model trained on domain data. These anomaly data should be provided with as much information as possible about the anomalies they contain. For instance, a log of ten minutes worth of flight data for a team of five autonomous drones could include information that specifies that drone #4 collided with a tree at time 7:15. This information is useful for checking that the anomaly detection system finds the specific anomaly being described.

Finally, as further described in Section V, the active learning module is dependent on expert input. We assume that limited expert time is available, and furthermore, that the data is such that the expert can diagnose the situation from the data.

V. Solution

We seek to create anomaly detection and precursor identification systems in the following way: First, obtain a source of data in which both nominal and anomalous states can be observed. Second, create an anomaly detection pipeline that can identify statistical anomalies and identify them as operationally significant (or not), with a justification. Third, to identify anomaly precursors so that the anomaly could be mitigated or entirely avoided. The combination of these technologies will make the anomaly detection system more reliable, trustworthy, and likely to give actionable information. Moreover, the anomaly detection system could be used to monitor an online system (in this case, a search-and-almost-rescue mission) to predict, diagnose, and fix issues as they appear. We describe the components of this anomaly detection system below.

A. Inductive Monitoring System

IMS [2] estimates the deviation from normal system operations. It operates in two phases. The first phase is the model-building (i.e., training) phase, where IMS builds a model of nominal operations. It takes as input numeric data vectors capturing system state; in our case, these are telemetry from the simulated drone, e.g. position, speed, objects detected, and the like. IMS clusters these vectors to build its model of nominal operations. Notably, IMS does not need any examples of anomalies to create its model. For the most part, IMS does not need any expert input, except for possible adjustment of data vector parameter weights and clustering arguments.

The second phase is monitoring the system for anomalies. In this phase, IMS takes as input the same sort of data vectors that it used to build its model, though here anomalous data may also be included. No further training of the model is done. Rather, a deviation score is estimated, representing how much the system's state deviates from expected behavior. This deviation estimate is the distance of the data vector from the clusters in IMS's model, either the closest point in the nearest cluster or a weighted average distance from the nearest clusters representing a designated number of training data points (k-nearest neighbors). The latter option was chosen for our results. IMS can also produce an analysis of parameter contributions to the deviation score, i.e., which parameters contributed most to the deviation score.

B. Meta Monitoring System

The Meta Monitoring System (MMS) is an add-on to IMS, developed to address some difficulties when using IMS. The first issue is interpreting the deviation scores. The deviation scores have a clear relative interpretation, namely that lower deviation scores are less anomalous, but the independent interpretation of a particular value is not clear. For instance, does a deviation score of 3 indicate an anomaly? The second issue is fleeting high deviation scores that often occur from data noise or unusual intermediate states. To address both issues, MMS produces a deviation score, based on IMS's output, that is more easily interpreted.

Like IMS, MMS has two phases: a model-building training phase, and a monitoring phase. MMS takes IMS's deviation scores as its only input and does not use the original data vectors. MMS primarily uses deviation scores from nominal data, but unlike IMS, it can also make limited use of results from anomalous data. It builds two models: one of nominal deviation scores and one of anomalous deviation scores, each consisting of a probability distribution of deviation scores. It uses a Markov chain Monte Carlo method to fit a Bayesian model of nominal deviation scores, which is further described below. MMS's model of anomalous deviation scores is the same as the nominal model, but with greater variance. Anomalous deviation scores, if available, are used only to fine-tune this variance; otherwise a default value is used.

In the second phase, incoming IMS deviation scores are passed to the learned model, and probabilities of producing the observed deviation score are calculated for both models. The probabilities are combined to create the MMS score m_n at time n, an estimate of the likelihood of an underlying anomaly, as:

$$m_n = \frac{P\left(x_n \mid x_{n-1}, s_n = \mathcal{A}\right) P\left(s_n = \mathcal{A} \mid s_{n-1}\right)}{P\left(x_n \mid x_{n-1}, s_n = \mathcal{A}\right) P\left(s_n = \mathcal{A} \mid s_{n-1}\right) + P\left(x_n \mid x_{n-1}, s_n = \mathcal{N}\right) P\left(s_n = \mathcal{N} \mid s_{n-1}\right)} \tag{1}$$

where x_n is the deviation score at time n and s_n is the unobserved state of the monitored system at time n, either nominal ($\mathcal{N}$) or anomalous ($\mathcal{A}$).

The probabilities of observing deviation scores are modeled as Rice distributions, which is the distribution of distances from the origin to a bivariate normal distribution with the same variance in both dimensions. The probability density function of the Rice distribution is:

$$f\left(x \mid v, \sigma\right) = \frac{x}{\sigma^2} \exp\left(\frac{-\left(x^2 + v^2\right)}{2\sigma^2}\right) I_0\left(\frac{xv}{\sigma^2}\right) \tag{2}$$

where v is the distance from the origin to the mean of the aforementioned bivariate distibution, σ is its spread, and $I_0\left(z\right)$ is the modified Bessel function of the first kind, with order zero [6]. This modeling choice is motivated by a simplified model of IMS (consisting of one relevant cluster) and data vectors (as two dimensional vectors whose subsequent values follow a normal distribution). However, MMS does not use a single Rice distribution, but many, using functions of the prior observation and learned parameters to estimate v and σ of Eq. 2. For the i^{th} observation, the v_i parameter is modeled as:

$$v_i = \theta x_{i-1}^{\eta} + b \tag{3}$$

where x_{i-1} is the prior observation (i.e., at time $i-1$) and θ, η, and b are learned parameters governed by their own prior distributions. Likewise, the σ_i parameter is modeled as:

$$\sigma_i = \phi x_{i-1} + c \tag{4}$$

where once again, x_{i-1} is the prior observation and ϕ, and c are learned parameters governed by their own prior distributions.

Both the nominal and anomalous models use the equations above, with the nominal model's underlying normal distribution's expectation closer to the origin (thus, deviation scores should trend downward) then the anomalous model. For the nominal model, θ and η are no greater than 1 and expected to be less, setting the expectation of downward trending scores. On the other hand, for the anomalous model, θ and η are always 1, centering the underlying normal distribution on the current observation, and as a consequence, deviation scores under this model are expected to increase.

To convert the MMS score to a binary classification, an arbitrary threshold of 0.5 is used to distinguish nominal from anomalous. When anomalous deviation scores are available, the anomalous distributions are adjusted to maximize performance on the training data, typically for accuracy, but MMS can also be trained to favor false positives over false negatives (and vice versa). MMS can also take as input IMS parameter contributions and translate those to parameter contributions to the MMS score.

C. Active Learning

IMS and MMS work together to characterize normal performance and identify when anomalies occur. However, there are shortcomings that may arise in practice. First, not all statistical anomalies are operationally significant, and too many false alarms degrade the utility of anomaly detection. This problem is exacerbated when false negatives have been reduced at the cost of increased false positives, to avoid missing important anomalies. Second, although IMS and MMS can justify their scores in terms of the observed data, these justifications can be difficult to comprehend.

Both of these issues can be addressed through active learning [3]. In our active learning approach, a two-class support vector machine classifier is used to distinguish operationally significant (OS) anomalies (true positives) from non-operationally significant (NOS) anomalies (false positives). As with IMS and MMS, our active learning approach has two phases, a model-building training phase and a monitoring phase. In the training phase, the active learning module takes input from a subject matter expert (SME) to train this classifier on the original numerical vectors describing the system state. For each iteration of the training process, the SME is presented a data point previously identified by IMS/MMS as anomalous, marks it as OS or NOS, and provides a rationale. If the anomaly is OS, the SME also marks a time window that shows exactly where in the data the anomaly occurred. The rationale is parsed and is used along with the data in the time window to build new features that are added to the data. Then, the active learning module retrains its classifier, using this new truth value along with all the previous ones. It repeats this process until a budget is reached, typically constrained by the SME's availability. Typically, only a small fraction of the data need be classified to achieve excellent performance. The rationales provided by the SME can also be reused as explanations for similar instances.

The classifier has to be initialized with some labels first with a healthy mix of OS and NOS anomalies. We first use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [7] to cluster the statistical anomalies found by IMS/MMS, based on the percentage contribution of each variable to the event being an anomaly. Thus, similar anomalies would be clustered together, with any anomalies that don't fall into any of those clusters being put in a separate 'outlier' group. We use our domain knowledge of what kind of OS anomalies can happen and how common OS anomalies are to pick one of the following initial sampling strategies:

1) DBSCAN outliers only:
    - Picks initial set only from the outlier group, which has points that DBSCAN could not put into any cluster and which are thus are outliers in the data based on the DBSCAN parameters.
    - Useful when most OS anomalies look different from each other, while NOS anomalies look sometimes similar to each other and sometimes are outliers. Some of the outliers would then be NOS, and some would be OS.
    - If OS points are not significantly rarer than NOS points or especially if NOS anomalies are a little rarer than OS anomalies, it might be worthwhile to go for strategy (3)—DBSCAN Clusters and Outliers.
2) DBSCAN Clusters Only:
    - Picks bootstrap set only from points that DBSCAN could put into a cluster
    - Useful when OS anomalies as well as NOS anomalies can generally be grouped into clusters because of similar behavior. So some of the clusters are OS clusters and some are NOS clusters, and therefore both get represented in the bootstrap set.
3) DBSCAN Clusters And Outliers:
    - Picks one point from each cluster, and then picks as many outlier points as there are clusters
    - Useful when NOS anomalies mostly have other NOS anomalies similar to them and thus can generally be put into clusters, while OS anomalies are generally different from each other. So the points picked from the clusters are generally NOS, and the points picked from the outliers are generally OS.

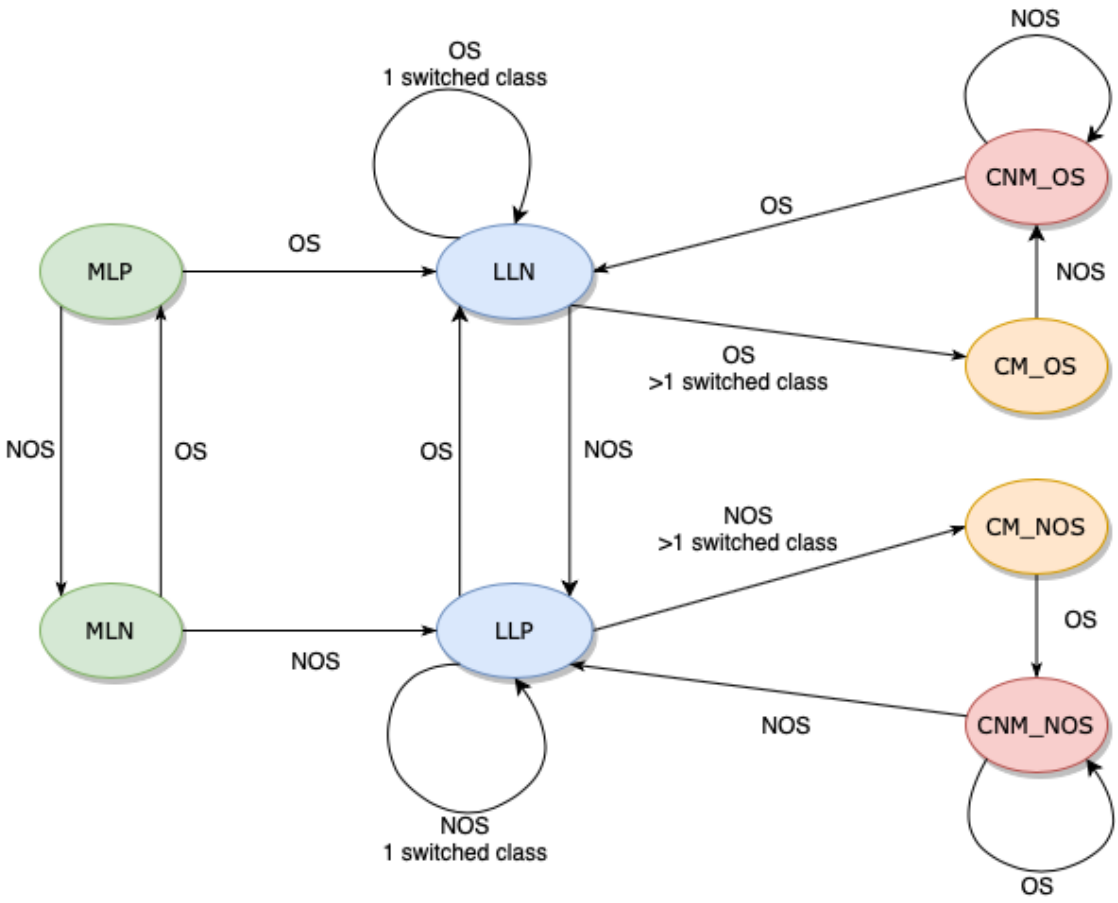

Fig. 3 Dynamic switching of Active Learning sampling strategies. The texts by the arrows are the label provided by the SME in the last iteration, and the arrows show the progression of strategies depending on the previous label, the previous strategy, and, in some cases, the number of other examples for which the label got switched due to updating the classifier.

The data point to be labeled at each iteration after the initial sampling is selected intelligently using various strategies that the model switches between dynamically based on model state attributes like what the previous label was and how many unlabeled examples switched classes after the last classifier update. Figure 3 describes how the model switches between strategies.

The active learning strategies that we use are:

- Most Likely Positive (MLP): Pick the data point that is farthest from the decision boundary on the OS side.
- Most Likely Negative (MLN): Pick the data point that is farthest from the decision boundary on the NOS side.
- Least Likely Positive (LLP): Pick the data point that is closest to the decision boundary on the OS side.
- Least Likely Negative (LLN): Pick the data point that is closest to the decision boundary on the NOS side.
- Check for Misclassified OS (CM_OS): If the last model update changed the label of more than one anomaly to OS, build a list of anomalies to check for potential misclassifications and pick the first one.
- Check for Next Misclassified OS (CNM_OS): Pick the next potentially mislabeled OS anomaly to check in the list.
- Check for Misclassified NOS (CM_NOS): If the last model update changed the label of more than one anomaly to NOS, build a list of anomalies to check for potential misclassifications and pick the first one.
- Check for Next Misclassified NOS (CNM_NOS): Pick the next potentially mislabeled NOS anomaly to check in the list.

D. DT-MIL

The last component of our anomaly detection suite is precursor identification. Identifying and explaining anomalies is advantageous, but even better if emerging anomalies can be identified and possibly mitigated. For this purpose, we use the Deep Temporal Multi-Instance Learning (DT-MIL) algorithm [5]. As with all our anomaly detection algorithms, DT-MIL has a model building (learning) phase and a monitoring phase. Like IMS, DT-MIL takes data vectors from the monitored system as input, but unlike IMS, these must be temporally sequenced since DT-MIL is time-oriented, as the name suggests. Also, not only can DT-MIL train using both nominal and anomalous data, it requires both. Thus, unlike IMS, DT-MIL cannot be applied to systems lacking anomalous data. In some cases where no anomalous data is available, the user may be able to derive a suitable substitute for actual anomalous data from available nominal data.

DT-MIL uses multiple-instance learning in a deep recurrent network to identify precursors. Multiple instance learning is a weakly supervised approach where instances are grouped into bags, but not all instances and bags are labeled (i.e., truth values are unknown). The algorithm is aided by the relationship between the bags and instances: typically, a bag is labeled positive if any of its contained instances are labeled positive, and is otherwise labeled negative. This fits well with precursor analysis where a sequence is known to contain an anomaly, but exactly where the anomaly develops is unknown.

DT-MIL uses a recurrent deep neural network to capture the progression of the anomaly and thus identify precursors. Specifically, DT-MIL uses gated recurrent units (GRUs) to capture the temporal relationship of successive instances. GRUs are similar to long short-term memory neural networks but have a simpler architecture which makes them easier to use. GRUs can transfer information from prior instances but can also inhibit this transmission, making them sensitive to the temporal characteristics of a particular anomaly.

E. An Anomaly Detection Pipeline

Figure 4 lays out the information flow in our anomaly detection pipeline. The pipeline includes all components except DT-MIL, which has its own information flow. The pipeline begins with IMS as it provides the initial characterization of anomalies. IMS consumes nominal, anomaly-free data covering normal system operation, labeled **a** in the figure, to build its model of nominal operations, labeled **b** in the figure. The model can then monitor a stream of operational data vectors, both nominal and anomalous, labeled **c** in the figure.

The IMS model produces both a series of deviation scores and corresponding parameter contributions. These are both passed to MMS, labeled **d** in the figure. MMS uses the deviation scores to build its model, which can then be used to monitor a stream of deviation scores. The resulting classifications and parameter contributions are then passed to the active learning module (labeled **e** in the figure), the last component of our pipeline.

The active learning module also receives the original data vectors (labeled **f** in figure), to build its classifiers and present to the SME (labeled **g** in the figure). The SME considers the event in question, provides a label and rationale to the active learning module (labeled **h** in the figure). The active learning module retrains its classifier with this new information, and the process repeats until some budget is reached, typically the time allocated to the SME. The final output (labeled **i** in the figure) are statistical anomalies flagged by IMS+MMS, and further refined to be marked as operationally significant (or not) by the active learning module, along with a corresponding rationale.

VI. Results

We used the Unity simulation described in Section III to generate the data needed to both train and evaluate our anomaly detection models. One hundred simulations were performed for each of our drone failure types. Within these simulations, we varied the size of the search area, number of drones, and individual drone operating characteristics. Using data with this variation helps to prevent our machine learning models from making their inferences based on trivially memorized details. This, in turn, makes the resulting models more likely to develop inference strategies that could generalize to test data, and to real world scenarios, where this level of variation would be present. The drones explore the search area in a "lawnmower" pattern, in which each drone moves across the environment in long, alternating lines from top to bottom, advancing a little to the right at the end of each line. As they move, these drones are also searching for a target that moves slowly across the ground. Should a drone fly over the target, it completes its search-and-almost-rescue mission and falls to the ground.

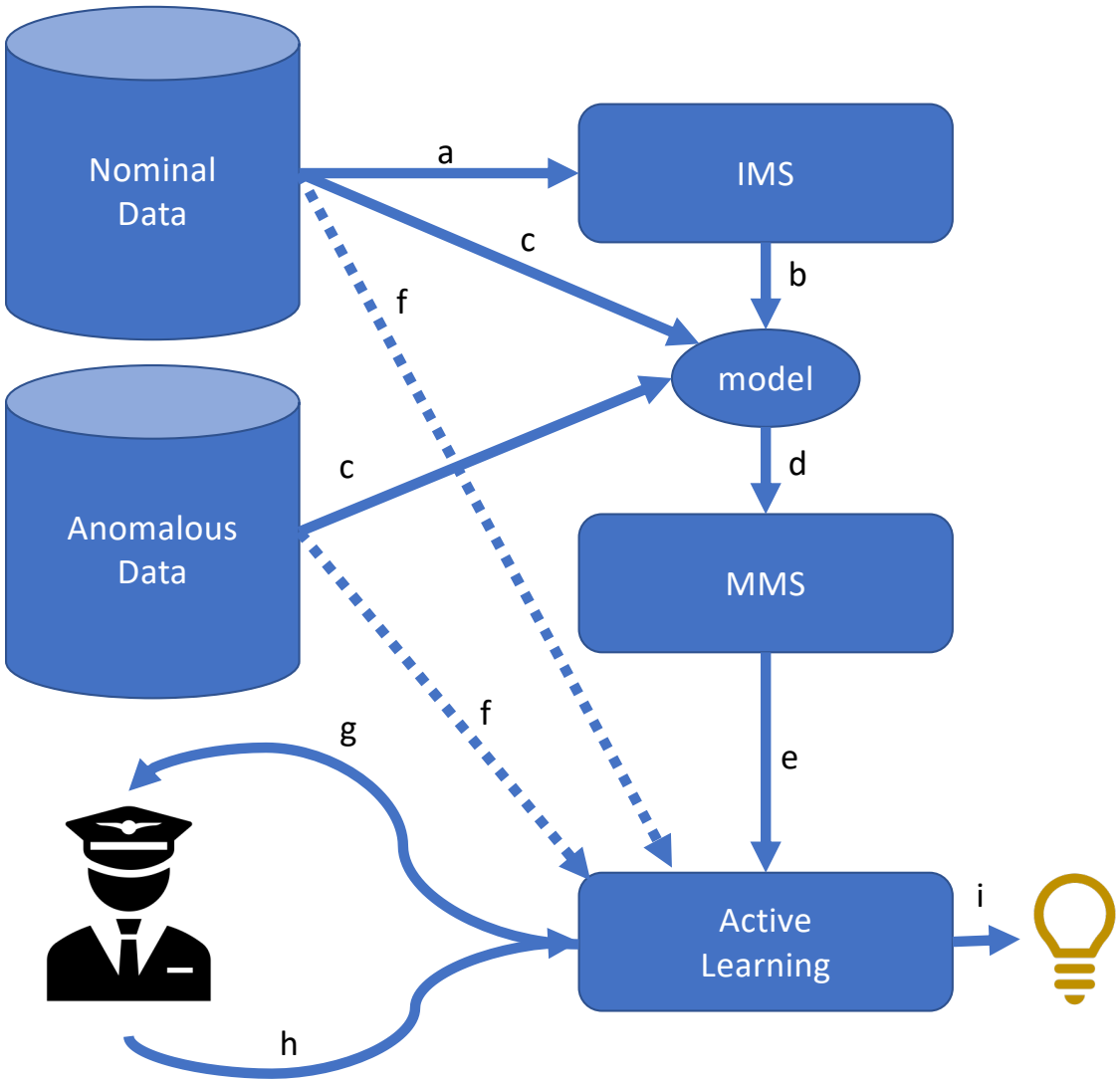

Fig. 4 Anomaly Detection Pipeline

A. Algorithmic

The performance of our algorithms are critically dependent on the data. as discussed in Section IV. The old adage of "garbage in, garbage out" generally applies to any data-driven method and ours are no exception. To some extent, performance is also dependent on any learning parameters, and in the case of IMS, user-supplied parameter weights will affect performance. Likewise, the quality of the SME's labels and rationales will also affect system utility.

B. Analytical

As a general rule, more data will improve performance but will experience diminishing returns. It is rare in real data to have completely separable classes, particularly given data noise, limited information, edge conditions, and ambiguous situations. IMS (and to a degree, MMS) are somewhat unusual in that they do not require examples of anomalies to build their models. For active learning and DT-MIL, some instances must be labeled, and performance may be negatively affected by class imbalance (many more instances of one type than the other, either nominal situations or anomalies).

C. Computational

We explored two failure scenarios in our simulations, covering both mechanical and operational failures:

- **Rotor failure**. In these scenarios, each of the four rotors of each drone has a random probability of degradation at each time step. These failures manifest as a loss of power to a rotor, and may be partial or complete. A rotor with less than 100% power generates less thrust than normal. A single rotor may experience more than one failure during a simulation. The drone controller will take normal course-corrective actions to get back on the flight plan, but will not otherwise compensate for any loss of power (i.e., the control algorithm will not stabilize the drone by changing the power of the other rotors). Loss of rotor power is permanent within each run of the simulation.
- **Obstacle avoidance**. In these scenarios, twenty trees of various sizes are randomly placed within the search area. These obstacles are stationary, but their location is not known to the drone operators ahead of time and are not accounted for in each drone's initial flight plan. Instead, a sufficiently close obstacle can be detected by the drone's LIDAR, giving the drone time to take evasive action. However, the drone's operator may fail to take evasive action and collide with the obstacle. This failure simulates a loss of situational awareness either from inattention or a sensor failure. The drone's operator regains awareness after a collision and will maneuver away from the obstacle, with no damage to the drone.

In addition, we fuse the two above data sets to create a new variant, **Novel**. The training set of **Novel** comes from the obstacle avoidance data set, whereas the testing set comes from the rotor failure data set. Thus, **Novel** explores the ability to detect an unanticipated anomaly: one that was not present in the training data. When analyzing each of the above data sets, we treat any rotor failure or crash as an anomaly, no matter how slight, and we do not quantify its severity. Using this data, we compare our methods with a variety of alternative approaches to anomaly detection. The total list of alternative approaches against which we compare is as follows:

- **Majority Class**. This anomaly detection method (also known as ZeroR) simply computes whether more simulations in the training data set were nominal or anomalous, then simply applies the resulting label to every piece of data it classifies. As such, this is not a legitimate anomaly detection method, but a means of finding the low bar for acceptable algorithm classification accuracy on this task. Any learning algorithm that has access to class labels should do approximately as well or better than this method.
- **Limit Check**. Perhaps the most common anomaly detection approach, limit checking defines a nominal operating range for each parameter of the available data. When classifying test data, this approach marks anything out of limits as anomalous. The limits for each data parameter can be defined by expert guidance or heuristically. In this case, we compute the limits for each parameter to be three standard deviations above or below that parameter's mean value. The mean and standard deviation of each parameter is estimated over the entire dataset.
- **IMS + MMS**. This is the first two stages in our anomaly pipeline, as described in Section V. The IMS model is built using the entire set of nominal data; none of the anomalous data is used. In contrast, ten-fold cross validation was used to tune the MMS model.
- **Active Learning**. This is our complete anomaly detection pipeline, including the IMS and MMS models and ending with active learning, as described in Section V. Active learning is trained iteratively on true and false positive instances and tested on the remainder of the data, and typically uses a small subset of the available data to train.
- **DT-MIL**. This is our precursor identification algorithm, described in Section V. Currently, DT-MIL randomly selects a training and validation set; we report the results on a held out testing set.

Table 1 Classification Accuracy by Flight

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	0.788	0.212	0.872	0.940	–	0.974
Obstacle	0.748	0.252	0.751	0.957	–	0.944
Novel	0.788	0.846	0.790	0.215	–	0.897

Table 2 Precision by Flight

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	1.00	0.212	0.672	0.824	–	0.946
Obstacle	1.00	0.252	1.00	0.935	–	0.927
Novel	1.00	0.582	1.00	0.215	–	0.969

We evaluate performance by classifying data by *flight* and by *event*. The task by flight is to analyze the entire recorded flight of a drone and classify it as positive if an anomaly occurs during the flight, negative otherwise. Event is more targeted, classifying flight segments that match an event of interest. For the obstacle data set, an event of interest is whenever a drone encounters a tree in its path, which it may either successfully avoid or collide with. For the rotor failure data set, an event of interest is anytime a rotors fails, which are all anomalies. For the nominal data of the rotor failure data set, we use a 10% sample of normally functioning drone (i.e., without rotor failure) flight segments of the same temporal length as the rotor failures.

We evaluate performance in terms of classification accuracy (*acc*), precision (*prec*), and recall (*rec*), given as:

$$acc = \frac{tp + tn}{tp + fp + fn + tn} \tag{5}$$

$$prec = \frac{tp}{tp + fp} \tag{6}$$

$$rec = \frac{tp}{tp + fn} \tag{7}$$

where tp are number of the true positives (correctly predicted anomaly), fp are the number of false positives (incorrectly predicted anomaly when the true class is nominal), fn are the number of false negative (incorrectly predicted nominal when the true class is anomaly), and tn are number of the true negatives (correctly predicted nominal). Since the goal of anomaly detection is to detect anomalies, anomalous is the positive class. As a consequence, since anomalies are less common, the majority class method suffers perfectly bad recall. For precision, we define $\frac{0}{0}$ as 1.

The performance of these algorithms in terms of our metrics can be seen in Tables 1-6. Overall, the majority class classifier gives a reasonable lower limit for accuracy, also indicating an approximate 3:1 nominal/off-nominal split for both data sets. Limit checker performance was computed by evaluating the performance of limit checks for each of the data parameters produced in these simulations. Table 1 reports a separate column of results for both the best and

Table 3 Recall by Flight

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	0.000	1.00	0.779	0.912	–	0.930
Obstacle	0.000	1.000	0.0110	0.890	–	0.845
Novel	0.000	0.973	0.00885	1.00	–	0.539

Table 4 Classification Accuracy by Event

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	0.762	0.626	0.901	0.988	–	0.992
Obstacle	0.707	0.304	0.802	0.834	0.933	0.908
Novel	0.762	0.626	0.759	0.896	–	0.535

Table 5 Precision by Event

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	1.00	0.280	0.983	0.966	–	0.983
Obstacle	1.00	0.253	0.927	0.897	0.860	0.857
Novel	1.00	0.280	0.417	0.592	–	0.221

worst performing of these limit checkers. For most parameters, limit checking yields accuracy below the majority class baseline and poor performance overall, with the best performing parameter yielding only fair results. Heavy tails in parameter distributions (including nominal runs) are part of the problem, but the primary issue is that extreme values occur in most nominal runs. As a result, limit checks tended to predict mostly a single class, either nominal or anomalous, depending on the parameter.

In contrast, each of our anomaly detection routines considerably outperforms the majority class baseline. The results for IMS+MMS and DT-MIL are comparable, with the strongest performer depending on the data set and the metric. IMS+MMS often performed worse than DT-MIL on the rotor failure data and better on the obstacle avoidance data, which is surprising as the rotor failure data set has no precursors—the failure occurs without warning. Apparently DT-MIL also functions well for anomaly detection in the absence of precursors, and IMS+MMS can detect emerging hazardous conditions like a failure to maintain adequate obstacle separation. Likewise, we had expected IMS+MMS to perform better on the novel anomaly, but this was only true for events and not entire flights. In the latter case, IMS+MMS consistently mistook regular turns as unusual situations and ended up flagging the entire (test) data set as anomalous. Turns have been the most consistent source of false positives throughout this work, and this particular defect will be addressed in future work.

The Active Learning model significantly boosts performance over IMS+MMS alone by considerably increasing recall with a very small loss in precision after labeling only 30 events. While training the model, whenever an event looked like an event of interest to the SME, the SME's label was provided to the model, even if it contradicted ground truth. The model proved to be robust against occasional false labels, getting high precision and recall when evaluated against ground truth labels. Active Learning had the best performance overall on the data set to which it was applied.

VII. Conclusions

In our work, we successfully demonstrated that data-driven anomaly detection with limited domain expert feedback on the operational significance of identified anomalies can effectively detect operationally significant anomalies during operations and provide explanations. We demonstrated that data-driven precursor identification methods allow domain

Table 6 Recall by Event

Dataset	Majority Class	Worst Parameter	Best Parameter	IMS + MMS	Active Learning	DT-MIL
Rotor	0.00	0.362	0.593	0.951	–	0.962
Obstacle	0.00	0.704	0.352	0.284	0.828	0.725
Novel	0.00	0.362	0.0345	0.973	–	0.835

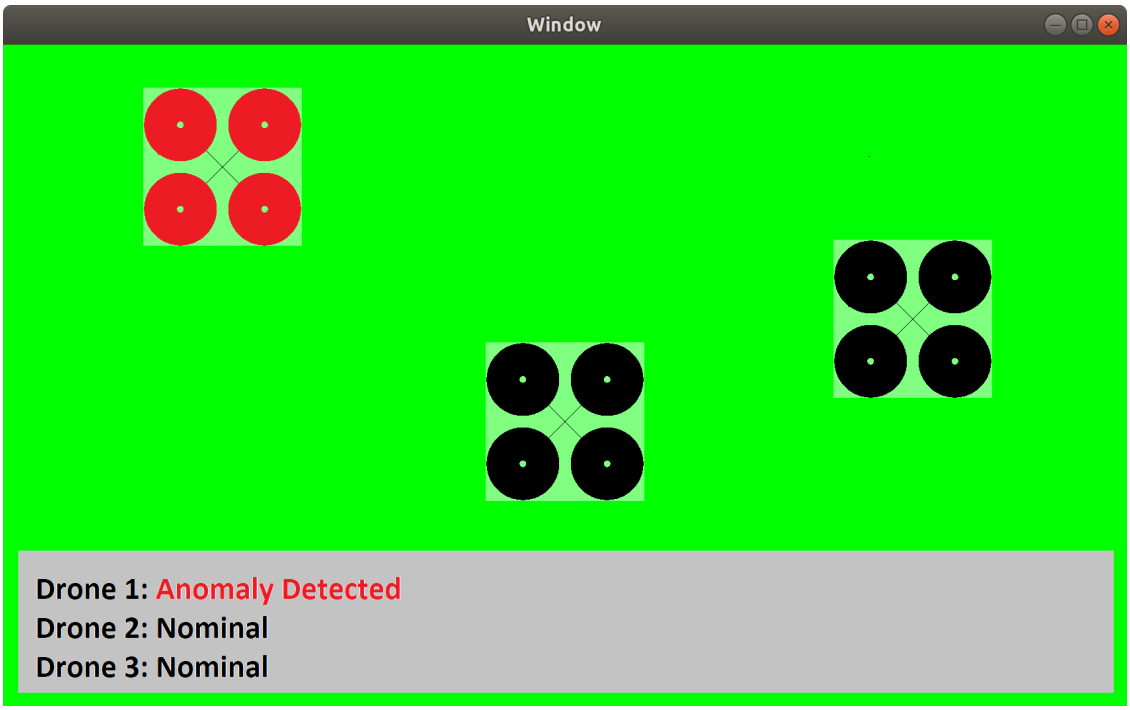

Fig. 5 Example of an online anomaly monitoring display

experts to identify undesirable effects and automatically identify their precursors. To that end, we feel that we have demonstrated one example of effective teamwork between human domain experts and artificial intelligence (specifically, machine learning) to identify sequences of events that lead to anomalous operations. Such teamwork can bring about increased trust in autonomous systems.

Our greatest difficulty in this work is matching our algorithms with the data. Preparing demonstration datasets with sufficient normal and anomalous behavior turned out to be remarkably difficult. Additionally, as with all other machine learning problems that we have worked with, a significant amount of time must be spent understanding the data and making adjustments to the machine learning algorithm(s) so that they work effectively for the task.

VIII. Future Work

We plan to work on setting up our algorithms to perform real-time monitoring during multi-UAS missions. In particular, we plan to develop software that can display the output of anomaly detection performed in real time during an operation for the benefit of individual UAS operators or mission managers. Assuming our familiar example of monitoring a collection of drones as motivation, this software should make it easy to determine which drones have been flagged as potentially experiencing an operationally significant anomaly at any time. It should also provide an indication of other available and relevant information, e.g. the last reported locations of each drone, although the exact nature of this information would have to be developed based on the requirements of an actual user. In order to illustrate what this software could look like, a small tool was developed which can parse information reported about a list of drones in real time and update an animated display with this information. Figure 5 shows the output of this software, and provides some indication of what an actual application's user interface might include.

Our anomaly detection pipeline used active learning to separate the statistical anomalies found by IMS and MMS into operationally significant (OS) and non-operationally significant (NOS). We could reconfigure the pipeline to have DT-MIL perform the role of IMS and MMS, even though DT-MIL's aim is different from that of IMS and MMS. We could also reconfigure the pipeline so that there is a feedback loop, with the output of the active learning module being used to refine models earlier in the pipeline.

Finally, the component elements of our anomaly detection suite could benefit from further enhancement. IMS is rather mature, but nonetheless would benefit from a module to assist in automatically tuning its hyperparameters. Additionally, it was found during early work using IMS for the multiple-vehicle monitoring task that enhancements to

the IMS anomaly metric calculations to incorporate more individual parameter behavior information could improve detection results. This will be investigated as we explore the multi-vehicle space further. MMS's training routine is rather slow and might be sped up with different techniques, such as variational inference [8]. Several active learning strategies for picking instances for SME review are currently being refined. DT-MIL has a nascent parameter precursor analysis that could be useful for explanations with further development.

References

[1] Alexandrov, N., and Allen, B., "Proceedings of the ATTRACTOR Feasibility Study," Tech. Rep. NASA TM 2021-XXXXX, to appear, National Aeronautics and Space Administration, 2021.

[2] Iverson, D. L., "Inductive system health monitoring," *In Proceedings of The 2004 International Conference on Artificial Intelligence (IC-AI04), Las Vegas*, CSREA Press, 2004.

[3] Sharma, M., Das, K., Bilgic, M., Matthews, B., Nielsen, D., and Oza, N., "Active Learning with Rationales for Identifying Operationally Significant Anomalies in Aviation," *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, 2016. URL `http://www.cs.iit.edu/~ml/pdfs/sharma-ecmlpkdd16.pdf`.

[4] Das, K., Avrekh, I., Matthews, B., Sharma, M., and Oza, N., "ASK-the-Expert: Active Learning Based Knowledge Discovery Using the Expert," *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*, 2017.

[5] Janakiraman, V. M., "Explaining Aviation Safety Incidents Using Deep Temporal Multiple Instance Learning," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, edited by Y. Guo and F. Farooq, ACM, 2018, pp. 406–415. https://doi.org/10.1145/3219819.3219871, URL https://doi.org/10.1145/3219819.3219871.

[6] WikiMedia Foundation, "Rice distribution," , 2020. URL https://en.wikipedia.org/wiki/Rice_distribution,accessedonFebruary6, 2020.

[7] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A density-based algorithm for discovering clusters in large spatial databases with noise," AAAI Press, 1996, pp. 226–231.

[8] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed., Springer, 2007. URL http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738.