Machine Learning Service for Vulnerability Discovery

Daniel Weckler* and Bryan Matthews †
Data Sciences Group, NASA Ames Research Center, Moffett Field, CA 94035, USA

I. Extended Abstract

Stakeholders such as airlines, the FAA, and airports aim to maintain safe and efficient operations in the National Airspace System (NAS). To address this need, the stakeholders have Safety Management Systems (SMS) in place that track key performance metrics and safety events over time. These programs are well established and provide critical insights into safety events that can lead to actionable changes to the operations to improve safety. However, there is a need to identify emerging issues that may not be tracked under the current programs. Without identifying these vulnerabilities, system blind spots to unknown risks may exist, and if not addressed, could lead to an incident or accident.

The purpose of this project is to provide a service that identifies unusual traffic patterns potentially related to safety or efficiency, supporting an in-time aviation safety management system (IASMS) concept [1]. Additionally, these traffic patterns may point to increased complexity in the airspace which highlight when air traffic controller workload may be elevated. The service provides an opportunity to perform vulnerability discovery to identify the "unknown unknowns" — in other words, draw attention to what is not currently being monitored with current SMS programs. Finally, this gives us the opportunity to monitor changes in the behavior of the overall airspace over a long period of time. Sudden or increased anomaly trends may point to a significant change in the system that deviates from the historical traffic patterns. This does not always indicate that there is an increased safety concern. However, the increasing anomaly trend means that there is an increasing deviation from the historical traffic patterns. Stakeholders may expect this due to new procedural changes going into effect or it may be unexpected, indicating that there is a need to investigate further. This service enables stakeholders to have more visibility into vulnerabilities so that proper responses can be taken to ultimately improve the safety of the NAS.

We have two data sources that enable our analysis: NASA's Sherlock Data Warehouse [2], and the FAA's System Wide Information Management portal (SWIM) [3]. Sherlock is a historical archive of already processed flight track data and flight information starting in 2017 and continues to update daily. SWIM is a livestream source that the FAA has made directly available to the public. We use our system to capture the SWIM data and store it for later use. The architecture for this system has been developed in the Python programming language with an SQLite backend for the database.

While Sherlock gets its data from the same source as SWIM, the data is archived in a different format than the SWIM livestream data. With some adjustments, reasonable parity between the two data sources can be achieved. Since the SWIM data feed will need to be run for months to build up a significant training data set, we leverage Sherlock data to initially bootstrap our model training. As more SWIM data is collected, the goal is to transition from training a model on only Sherlock data, to training on a hybrid of both data, and eventually train on only the collected SWIM data. An example of the pipeline is shown in Fig. 1. The data capture process involves connecting to the SWIFT (SWIM Industry-FAA Team) portal [4] and streaming the flight track data from the Terminal Automation Information Service (TAIS) for the Northern California TRACON (NCT). The timestamped data captured from this source includes latitude, longitude, speed, altitude, and destination airport. The flights are identified by an Aircraft Identifier (AcId), otherwise known as a callsign. A second data capture process monitors the "R14 Flight Data" stream, which includes "estimated" and "actual" landing times for a flight. Each individual flight track sample is stored in a "live flights" database. When an "actual" landing time is detected, the flight data is pulled from the database, processed, and saved to disk. The processed flight data is then fed to the model, which returns an anomaly score. The landing information, in addition to the anomaly score, are then output to a "landing info" database that is accessible via a RESTful API that is integrated with the NASA's Digital Information Platform (DIP) service. DIP is a cloud based information platform that aims to assist in decision making for airspace operations. It provides a variety of real time services that stakeholder users can subscribe to and integrate into their existing systems with API calls [5]. There are two main advantages to using this platform to serve this data product: 1) the users' credentials are managed by DIP, reducing the security management overhead needed to ensure secure interactions with a public facing API and 2) the platform has attracted an existing

*Research Engineer, KBR, Inc.
†Senior Data Science Research Engineer, KBR, Inc.

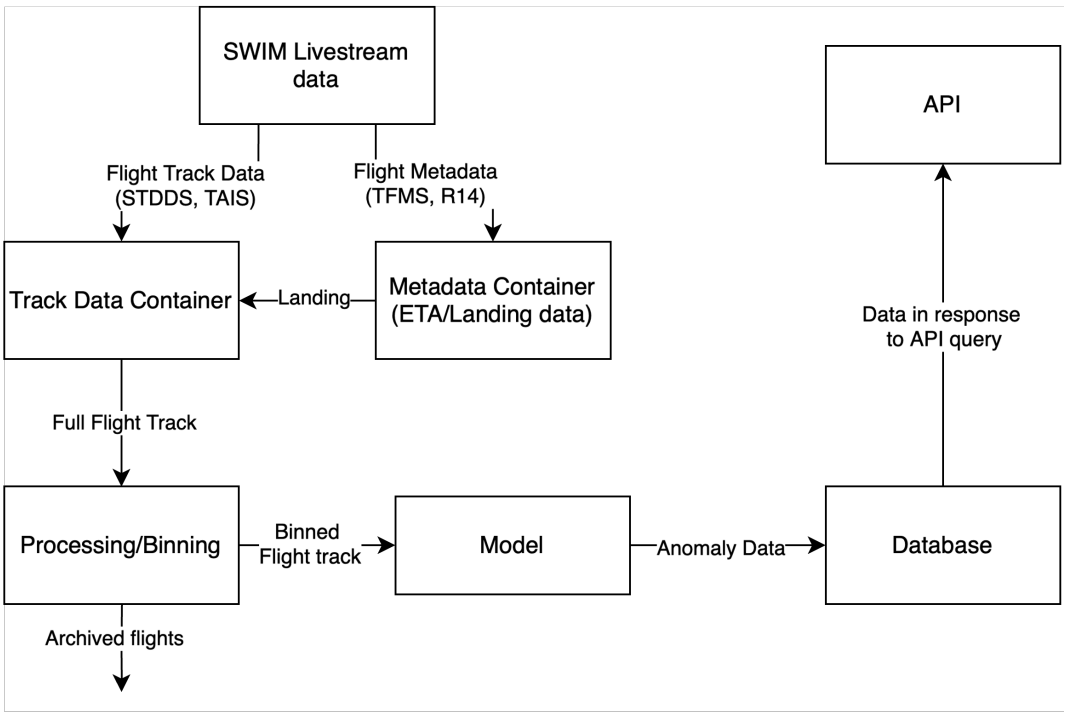

Fig. 1 Data Capture Pipeline

pool of industry and government stakeholders that have an interest in utilizing DIP services to improve their operations, giving the service visibility within an environment stakeholders are familiar with.

A significant amount of data processing is done on the captured data. During processing, the data is converted from a time series to a "distance series." This series is comprised of the last 30 miles of flight data with ¼ mile bins. An example of this for SFO is shown in Fig. 2. This is done so that the algorithm observes the final 30 miles of approach with equal vector lengths for all flights. Several more parameters are computed during data processing. These derived parameters include: "vertical speed" (rate of change of altitude), radial distance to the airport, and angle of the flight relative to the airport. For more efficient model training, the flights are also separated by airport as well as each individual runway in that airport. Even for individual runways, the final 30 miles of each flight's approach can differ greatly. To our knowledge there's no parameter in the SWIM data that defines these unique transition paths, so we had to take an additional step. To help the algorithm more accurately distinguish anomalous flights, we utilize K-means clustering to separate flight paths into their own clusters. We use the flight's start point (at the "beginning" of the 30 mile path) and the bearing relative to the airport as features for the clustering algorithm. To mitigate data quality issues caused by low sample sizes, we combine clusters that have less than 1000 flights in their training data. These flights effectively become a background cluster that we do not expect to perform well upon reconstruction. However, the statistical threshold that is computed that determines whether a flight is anomalous or not will be based on this cluster's reconstruction ability and therefore the background cluster will not proportionally be higher than other clusters.

We utilize a neural network model called Convolutional Variational Auto-Encoder (CVAE) [6] to detect anomalies. CVAE is an unsupervised encoder-decoder model for anomaly detection in multivariate time-series. At a high level, it takes an input time series, learns a model to encode the time series into a compressed space, and reconstructs it. The more examples of flight data it has, the better the model is at reconstructing. In theory, anomalous flights are rarer, so they will be more difficult for CVAE to reconstruct. As the model reconstructs the original timeseries input we measure the error between the model reconstruction and the original input space. The assumption is that flight track patterns that are unusual or "rare" will have higher reconstruction errors.

A validation dataset is set aside to compute the statistical quantities such as mean and standard deviation for each of the conditional categories and mean + 3 * standard deviation is used as a threshold to detect flights of interest — including anomalous flights. The version of CVAE we use for the project is called CCVAE. CCVAE stands for Conditional Convolutional Variational Auto-Encoder. It is an expansion on CVAE. but it takes a conditional input to

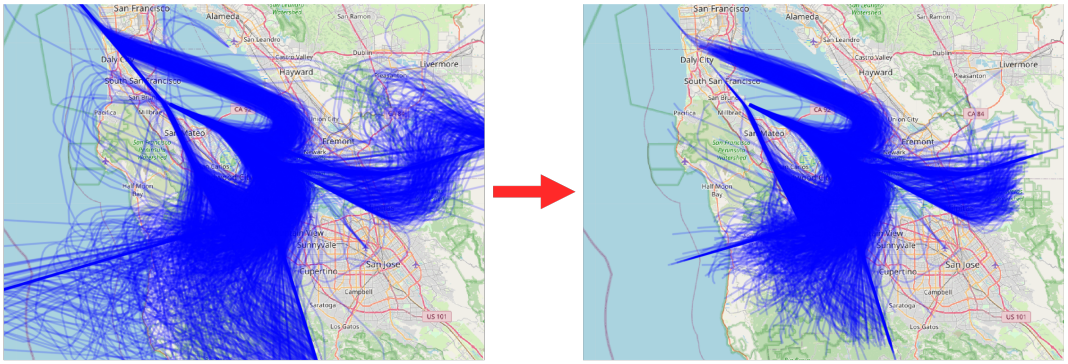

Fig. 2 Binning the flight track data to the last 30 miles

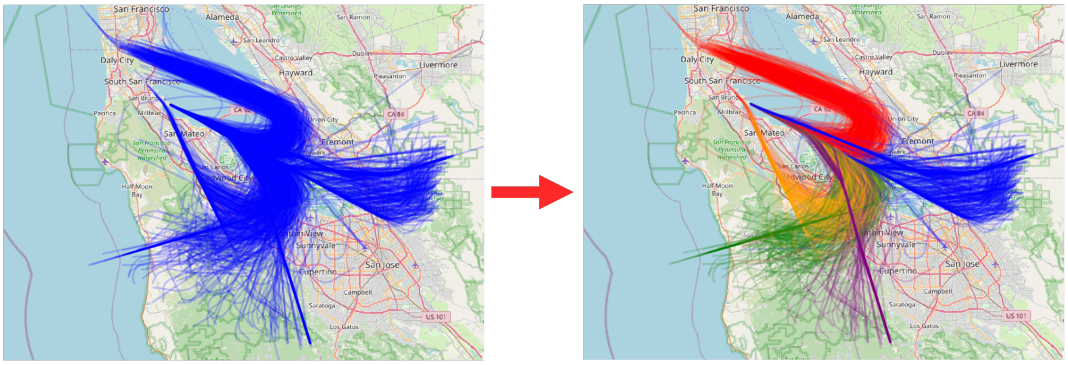

Fig. 3 The results of using K-means to cluster the data

map data with similar conditionals to a unique space in the model's latent space. This results in one global model that is similar to utilizing multiple models. The categorical split allows the model to better learn the specific flight patterns and other flight data unique to each airport, runway, and traffic-flow (categorized during clustering) combination. The more homogeneous the traffic flow patterns are to each runway and airport, the more the distribution of the reconstruction error scores will resemble a Gaussian distribution and therefore a mean + 3 * standard deviation threshold is appropriate for our anomaly score threshold, as seen in Fig. 4. Flights that score below that threshold are considered "nominal," and flights above that threshold are our "anomalous" flights. Each airport/runway/traffic-flow-cluster combination will have its own distribution and anomaly score.

We utilize Python's FASTApi package to integrate with NASA's DIP service to provide in-time results on captured data. This API offers two main endpoints: model training parameters, and flight data/predictions. The model training parameters endpoint returns a JSON packet that contains a variety of information on the current model being used to detect anomalies. This data includes: the number of flights the model was trained on, the dates the model was trained on, all the categorical parameters used in training (airport/runway/cluster), training hyperparameters, and calculated reconstruction errors for each combination of categorical parameters used in training. The other endpoint allows the user to request data from flights throughout the processed flight database. This includes callsign (AcId), landing time, landing runway, the anomaly score, and whether or not a flight is considered an anomaly. The user can query flights on a specific day, all flights on an airline, specific month, and more. Fig. 5 shows an example of a data packet that can be sent to the API.

Stakeholders may use the data collected from the API to monitor temporal anomaly trends and patterns across different demographics by airport, runway, or airline. Conceptually, the API can be tied into a dashboard that reflects in-time anomaly statistics and augment an existing SMS program. For example: Fig. 6 shows the rate of runway anomalies over a four month period color coded by airport. This may help inform which runways have elevated proportions of anomalies as compared to others in the same airspace or airport. Fig. 7 illustrates how the the rate of anomalies by airport changes proportionally over the the same time period. Monitoring these changes over time indicates where unusual flight track behavior is increasing or decreasing and may correspond to known operational

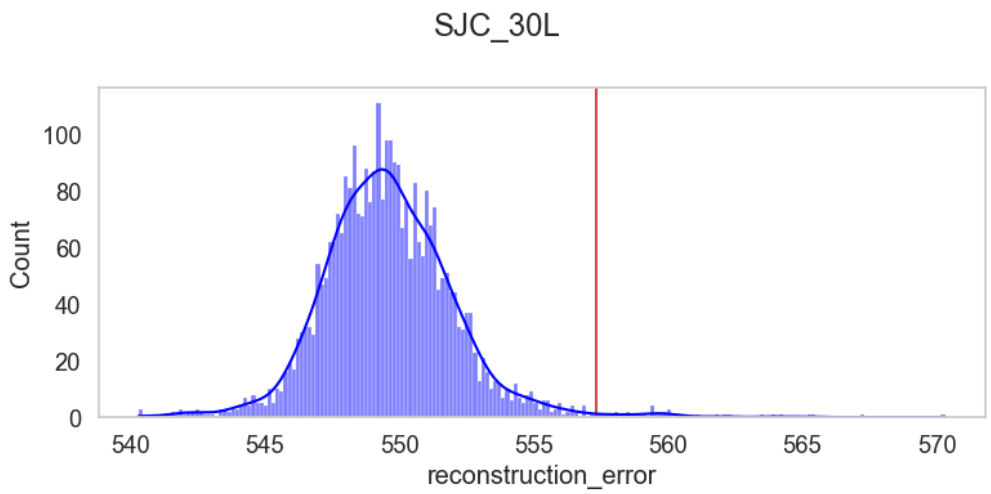

Fig. 4 An example of the distribution of reconstruction errors for runway 30L at SJC. The red line is the threshold indicating anomalous reconstruction errors.

```
{"acids": ["UAL*", "SWA*"], "dates": ["2024*"]}
```

Fig. 5 In-context JSON schema for a data packet.

changes or require additional context to explain the changes. This capability offers new ways to monitor flight track behavior and provides awareness to changes in airspace flight patterns as compared to historical flight tracks. Leveraging this data can inform and assist decision making for IASMS programs and is a key component to the SMS life cycle.

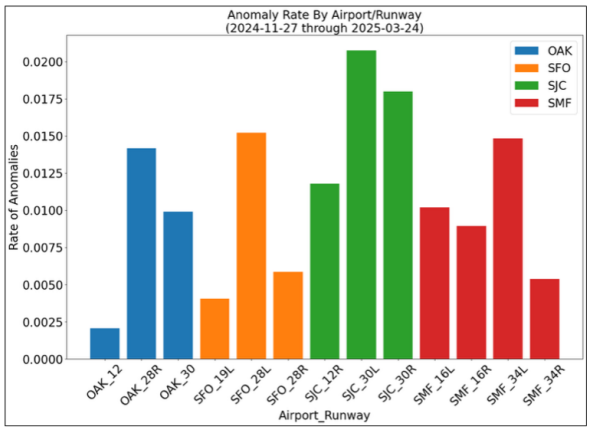

Fig. 6 Anomaly rates by airport and runway.

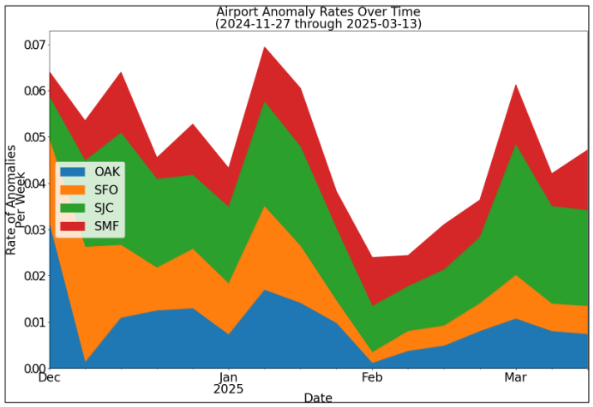

Fig. 7 Rate of anomalies by airport over time.

References

[1] Ellis, K. K., Prinzel, L. J., Vincent, M. J., Krois, P., Ackerson, J., Infeld, S. I., Brandt, S. L., Okolo, W., Coughlan, J., Davies, M. D., Mah, R., Oza, N., Stephens, C. L., and Glenn-Chase, A., "An In-Time Aviation Safety Management System Concept of Operations and Modernization of the National Airspace System," , 2025. https://doi.org/10.2514/6.2025-2555, URL https://arc.aiaa.org/doi/abs/10.2514/6.2025-2555.

[2] Eshow, M. M., Lui, M., and Ranjan, S., "Architecture and capabilities of a data warehouse for ATM research," *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, 2014, pp. 1E3–1–1E3–14. https://doi.org/10.1109/DASC.2014.6979418.

[3] Meserole, J. S., and Moore, J. W., "What is System Wide Information Management (SWIM)?" *2006 ieee/aiaa 25TH Digital Avionics Systems Conference*, 2006, pp. 1–8. https://doi.org/10.1109/DASC.2006.313756.

[4] Federal Aviation Administration (FAA), "SWIM Industry-FAA TEAM (SWIFT)," https://portal.swim.faa.gov, 2025.

[5] Gurram, M. M., Hegde, P., and Saxena, S., *NASA's Digital Information Platform (DIP) to Accelerate NAS Transformation*, 2023. https://doi.org/10.2514/6.2023-3400, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-3400.

[6] Memarzadeh, M., Matthews, B., and Avrekh, I., "Unsupervised Anomaly Detection in Flight Data Using Convolutional Variational Auto-Encoder," *Aerospace*, Vol. 7, 2020, p. 115. https://doi.org/10.3390/aerospace7080115.