Proceedings of the 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October, San Antonio, TX, USA
DOI: 10.1109/DASC52595.2021.9594321

Enhancing the Security of Unmanned Aerial Systems using Digital-Twin Technology and Intrusion Detection

Benjamin Fraser, Saba Al-Rubaye, Sohaib Aslam, Antonios Tsourdos
School of Aerospace, Transport and Manufacturing, Cranfield University
Milton Keynes, United Kingdom
Email: {b.fraser, s.alrubaye, s.aslam, a.tsourdos}@cranfield.ac.uk}

***Abstract*— In this paper the general susceptibilities of Unmanned Aerial Vehicles (UAVs) against modern-cyber threats are explored and potential solutions proposed. This is achieved by applying digital-twin architectures and data-driven methods to UAVs to facilitate identification of real-time intrusions and anomalies. These concepts are validated by performing novelty detection on open access UAV flight data with GPS spoofing attacks, which represents a typical system use-case. Multiple machine learning models are trained to demonstrate the feasibility of detecting modern cyber-intrusions and anomalies using the digital-twin architecture. This includes both classical and deep learning techniques to help identify the most suitable model types for the proposed design. The overall results are positive and help highlight the potential of digital-twin architectures for the UAV contexts.**

Keywords—Digital-twin, Machine Learning, Novelty Detection, UAV, UAS, Intrusions.

I. INTRODUCTION

The adoption of Unmanned Aerial Systems (UAS) has grown rapidly in recent years, with established use-cases throughout both civilian and military domains. This includes many high-risk and safety-critical applications like military operations, structural inspection, and emergency service response. In these contexts there are serious consequences if security is compromised, especially through vulnerabilities or malicious cyber-attacks.

The number of attacks on cyber physical systems has increased significantly based on publicly available information [1]. Unmanned Aerial Vehicles (UAVs) rely critically on external sensor data such as Global Positioning System (GPS), and this reliance presents vulnerabilities to attacks or exploits like spoofing and manipulation. These attacks are feasible at low-cost using off the shelf components and can be targeted against high-value assets, such as aircraft, maritime vessels and UAVs [2]. These malicious actions can lead to unintended control movements, collisions, crashes and hijackings.

The rapid emergence of UAV technology and the use of concepts from manned aviation means most designs did not originally centre around cyber-security and safety [3]. Subsequently, many critical sub-systems can be susceptible to physical and cyber-based attacks. Therefore, the ability to detect intrusions and mitigate their impact is important for flight safety and mission success. Existing studies have explored UAV vulnerabilities, including [3] and [4], who emphasise the key issues associated with navigation, communication and control-based attacks. Anomaly detection techniques are effective for countering these threats and many recent advances in this field can be applied to aviation [5]. Supervised learning-based models show promise for detecting intrusions when labelled data is available, including GPS-spoofing [6], ADS-B attacks [7], Inertial Measurement Unit (IMU) malfunctions [8] and network intrusions [9]. Sufficient quantities of labelled anomalies are difficult to acquire however, and so unsupervised learning overcomes this by modelling normal behaviour to identify anomalies without the requirement for labels. This is known as novelty detection and is well-studied for UAVs, with classical-based approaches like One-Class Support Vector Machine (OC-SVM), Isolation Forest (IF) [10] and Local Outlier Factor (LOF) [11] being common. Recently, deep unsupervised learning architectures have emerged, including autoencoders for automatic UAV fault detection [12] and Long-Term Short-Term (LSTM) networks for detecting ADS-B manipulation [7]. These generally show significant performance gains compared to classical techniques, at the expense of increased complexity, computational overhead and data quantities.

Digital-twin architectures enable development of high-fidelity models for real systems using vast quantities of operational data and expert insights. For health monitoring and anomaly detection, such architectures can identify intrusions or anomalies that fall outside normal operation. Existing literature has explored this for industrial applications using similar techniques to those above, including [13], [14], and [15] who use digital-twins to detect anomalies in manufacturing and energy systems using machine learning. The concepts and benefits realised should equally apply to UAVs.

In this paper, UAS architectural enhancements are proposed to reduce the impacts of sensor-based security threats and anomalies during flight. This is achieved through applying digital-twin architectures and data-driven methods to UAV platforms. This develops the concepts from existing work on digital twins [13] and applies it to aviation for real-time UAV novelty detection of intrusions using machine learning. The proposed design is validated by evaluating various novelty detection models on UAV data with GPS sub-system spoofing attacks. These models are representative of how the digital-twin would represent normal behaviour across each sub-systems and use this to detect real-time malicious intrusions or anomalies. Discussions follow the analysis on the most suitable model types to be used. In the true system, these would extend beyond just GPS sub-system monitoring.

The aim of this paper is to improve the cyber-security, operational safety and reliability of UAVs through applying data-driven methods and a digital twin architecture to complement human-in-the-loop systems. The contributions, in addition to existing studies, are the following:

1) Application of digital-twin architectures to the context of UAVs for real-time monitoring of intrusions and anomalies to enhance security.

2) Application of real-time time-sequencing novelty detection machine learning models to UAV flight log time-series data.

II. Technical Background

A. Unmanned Aerial Systems Overview

UAS architectures vary in size, shape and type. They are flown either autonomously through on-board computers, or remotely by an operator. Both of these options are available in some systems, subject to requirements. In general, there are three main aspects to a UAS [16]:

1) One or more UAVs containing mission-specific payloads.
2) A Ground Control Station (GCS) for UAV mission planning, programming, guidance and control.
3) Communications between system components.

This paper assumes that UAVs are programmed with mission plans from the GCS before flight. The UAV autonomously executes a mission using sensors, navigation, and control systems. The GCS maintains continuous communications, so that mission and UAV flight data is received in real-time. This also allows the operators to reprogram or reconfigure the onboard systems if required.

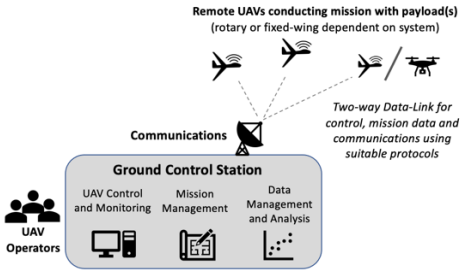

Fig. 1. High-level system interaction between GCS and remote UAVs.

B. Unmanned Aerial Vehicle Overview

The UAV is an embedded systems platform controlled and monitored by the GCS. The platform contains many technologies and sub-systems that perform sensing, communication, reasoning, actuation and control. Autonomous designs pilot themselves using embedded Flight Control Systems [17] that gather information from sensors, with a typical system shown in Fig 2.

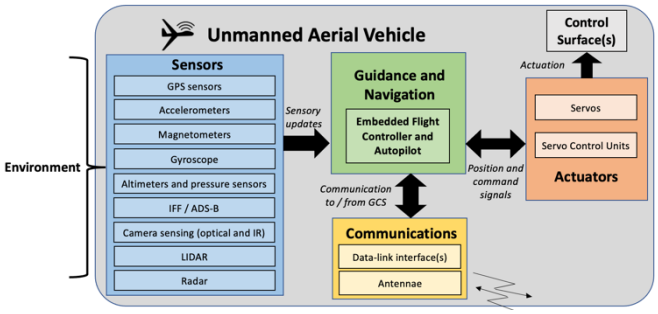

Fig. 2. Breakdown of major UAV sub-systems.

The sensors perceive from the environment and feed updates to the flight controller. The controller computes the required commands for the actuators to maintain safe and controlled flight, as configured from the GCS. This is conveniently framed as an agent-architecture, as shown in fig. 3.

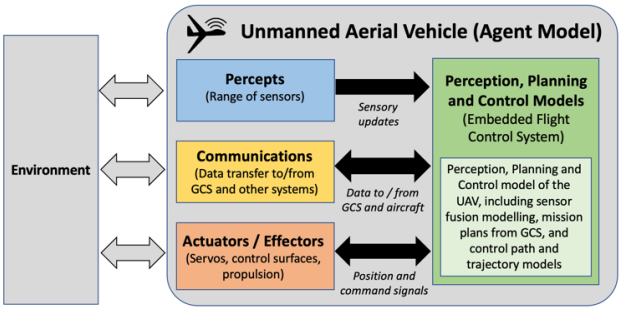

Fig. 3. UAV agent architecture model.

Especially important are the GPS and inertial navigation inputs such as accelerometers, altimeter, gyroscopes, speed-sensing, and magnetometers. These feed directly into the flight controller, and thus if inaccurate or maliciously tampered the consequences could lead to collisions, crashes or hijackings.

C. Ground Control Station (GCS)

The GCS is the interface through which the operators interact, monitor and control the UAVs. The design varies between systems, with examples being a smart device application, a remote hand-held controller, and a large-scale fixed command station [18]. Real-time data and commands can be sent between the GCS and UAV, and therefore the GCS represents a virtual cockpit. GCS and UAV wireless communications work using agreed protocols, subject to system type. This protocol forms a data-link between the system elements, thus allowing control commands and data to be transferred. Most systems make use of Line of Sight (LOS) communications, where the UAV stays within close enough range for direct radio waves. More complex systems may use Beyond Line of Sight (BLOS), where the UAV is controlled and monitored through satellite or relayed communications [19].

D. Security and Cyber-Threat Vulnerabilities

The Analysis of UAV threats by [3] highlights three main areas for potential attacks, with examples given in Table I:

1) Hardware attacks, where attackers have physical access to the UAV before flight.
2) Wireless attacks that exploit one or more of the system communications.
3) Sensor spoofing or jamming, where false information is sent to one or more sub-systems, leading to unintended control or system malfunctioning.

TABLE I. Attack Possibility Examples.

	Example 1	Example 2
Hardware Attacks	Fuzzing attacks on control command data-flows using implanted device or software	Manipulation of gain scheduling for control command inputs
Wireless Attacks	UAV Hijacking through breaking communication sub-system encryption.	Denial of service of UAV data-links, such as wide-band, narrow-band or Wi-Fi
Sensor Spoofing	Manipulation and malicious control of GPS positioning.	Spoofing or jamming of ADS-B system causing collisions or inadvertent movements.

Unless an attacker has physically exploited the UAV, the largest vulnerabilities are external attacks against sensors or communication protocols. The most common of these are spoofing or denial of service [18], which can cause loss of control, collisions, crashes and hijackings. Since UAVs rely significantly on sensor data for navigation and control, there are inherently many attack vectors [20].

For example, GPS systems operate using triangulation and a network of satellites that provide precise clock times and locations. After receiving signals from different satellites, a UAV GPS receiver can calculate its position in three-dimensional space [21]. Using off-the-shelf signal generators, the signals from real GPS satellites can be maliciously simulated and modified [2] to mislead UAVs of their true position, as shown in Fig. 4.

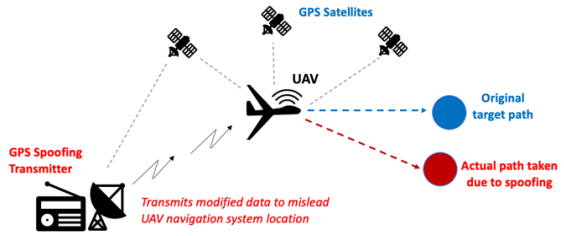

Fig. 4. Simplified diagram of a GPS spoofing scenario on a UAV.

Another example is Automatic Dependent Surveillance – Broadcast (ADS-B), which is a modern air traffic control system that transmits data such as location, airspeed, altitude, heading and aircraft identification data. This functions as a broadcast system to alert other aircraft and prevent collisions. UAVs with ADS-B monitor other aircraft, and if required, adjust flight to avoid potential collisions [3]. These transmissions may be falsified or spoofed to manipulate UAV flight, leading to the same issues as GPS.

E. Digital-Twin Technology

Digital-twin architectures are a growing trend and are increasingly feasible due to breakthroughs in big data, cloud computing and the Internet of Things [14]. They are digital representations of real systems, which model the behaviours and characteristics to improve the original entity, and have the potential to revolutionise most major industries [22]. The abundance of sensor-based data on UAVs makes them feasible for digital-twin modelling. In this paper, they will be applied for cyber-intrusion and anomaly detection. The results from such models could be used by the system or operators in real-time to perform follow-up safety measures or system reconfigurations to mitigate risks.

Conventional model-based anomaly detection is complex, time-consuming and often requires physical design aspects. It generally requires deep knowledge and system understanding, and therefore the production of many specialist models for all sub-systems can be impractical [5]. Data-driven techniques like machine learning provide suitable alternatives, as shown by [6] – [12], who perform UAV-based anomaly detection. These techniques will be used within the proposed digital-twin architecture.

III. System Design

A. Improved System Requirements

If the UAV could detect attacks this could improve safety, since mitigating actions could be used until the attack stops or the aircraft is recovered. Such actions could include disabling impacted systems, such as GPS, and instead relying on deduced reckoning from the other navigation sensors. It could also include alerting operators, so they can carefully monitor the UAV and abort flying if required. Thus, there is a need for effective UAV anomaly and intrusion detection, to facilitate operator-directed or automated mitigating actions. In this paper, communications and sensor-based attacks are focussed on. An improved design is proposed that mitigates the discussed safety concerns resulting from cyber-attacks on sensor and communications sub-systems.

This design will model normal flight behaviour and detect anomalies resulting from attacks. This will be accomplished by applying computational models to the UAV data in real-time and raising alerts or performing contingent actions when anomalies are detected. The GCS acts as the central hub for communication and control, and therefore any anomalies detected will be alerted to the operators for follow-up action or UAV reconfiguration.

B. System Design Description

The proposed design augments the existing system with a digital-twin architecture. This uses a digitised model for each UAV in the system, which learns and models the behaviour under normal and expected circumstances. The digital-twin uses a vast expanse of historic UAV flight data to achieve this. Overall, the proposed system contains two main components:

1) A digital-twin for each UAV that models normal behaviour using historic data.

2) A deployed monitoring component for real-time cyber-intrusion detection that is programmed into the UAV before flight.

A UAV digital-twin has many potential uses, including predictive maintenance, flight simulation, failure analysis, security enhancements and cyber-threat detection. For this work, only cyber-intrusion detection is considered. If a UAV sub-system is subjected to an attack during flight, the proposed system can detect and highlight this in real-time to enhance safety, reliability and robustness. It does this by discriminating between normal and anomalous flight behaviour.

Each digital-twin will contain multiple behaviour and novelty detection models trained on historic flight data from the expected environment. A model is produced for each major sub-system, which represents the expected sub-system behaviour and characteristics under normal flight conditions. This allows precise pinpointing of anomalies to a sub-system level. In addition to flight data, the system will benefit from the input of experts, including design engineers, maintenance engineers and UAV operators are depicted in Fig 5. These experts deeply understand the system and can provide insights into the model production process, in addition to the historic data.

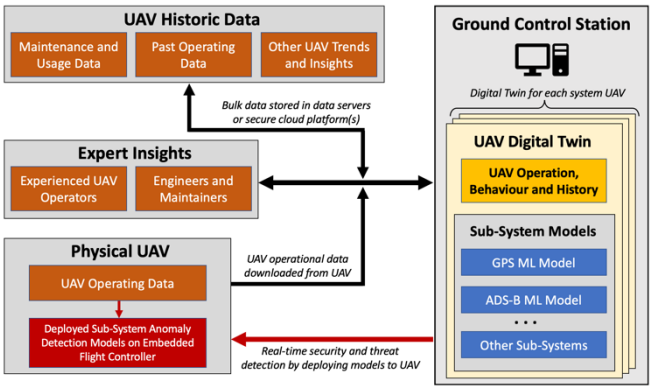

Fig. 5. Overview of the data interactions in the digital-twin architecture.

The UAV digital-twin is maintained on the GCS infrastructure. This allows the use of higher performance hardware and resources to exploit vast data quantities, which would be impractical on the UAV embedded systems. There may be times where the GCS and UAV have lost communications, or the latency is too high for real-time analysis, and so the UAV would require direct access to the models. Thus, the GCS will deploy trained model instances into each UAV before flight, during mission planning. This is the reason for the deployed monitoring component, which enables real-time inference of anomalies during flight.

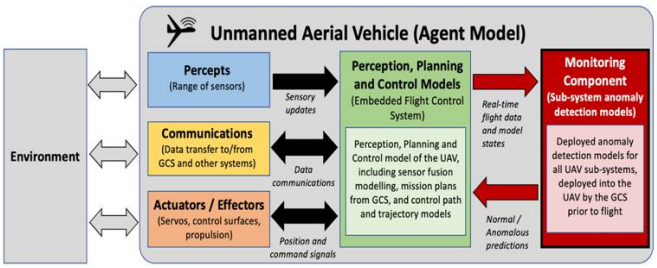

Fig. 6. UAV agent model with the deployed monitoring component.

The monitoring component is programmed into the flight controller and accesses real-time UAV flight features to continuously monitor for anomalies, which are fed back to the operators, as shown in Fig.6. This is similar to the work conducted by [3], who introduced the concept of an autopilot cyber-security supervisor. On detecting a sub-system anomaly the flight controller alerts the GCS. Depending on the operator pre-configured UAV settings, the flight controller could automatically reconfigure the system to mitigate the issue. For example, on detecting a GPS spoof attack, the flight controller automatically ignore GPS and rely on inertial navigation until the attack has stopped. This could be performed similarly on any other sub-system, provided it does not jeopardise safety as informed by experts in the design process.

These types of system configuration depend on the mission context and location, including safety and risk considerations, and therefore it would be an operators responsibility to configure the UAV appropriately. For example, if the environment was known to be routinely exposed to GPS spoofing attacks, it could be prudent to configure the flight controller to temporarily ignore suspect GPS-signals when an attack is detected, and instead rely on the inertial navigation sensors. Fig 7 shows the overall process for the proposed design, including the digital-twin and deployed component.

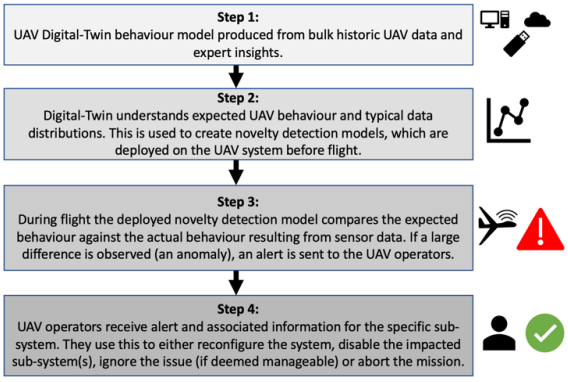

Fig. 7. Overview of the digital-twin modelling process for anomalies.

C. *GPS sub-system example*

To demonstrate the use of the proposed system, the example of GPS spoofing attacks will be used. It is assumed that the UAV GPS sensors extract a range of features from external GPS signals, such as satellite identification, longitude, latitude, altitude, signal strength, doppler shift and carrier phase characteristics. This, in addition to other sensor and navigation data is processed by the flight controller.

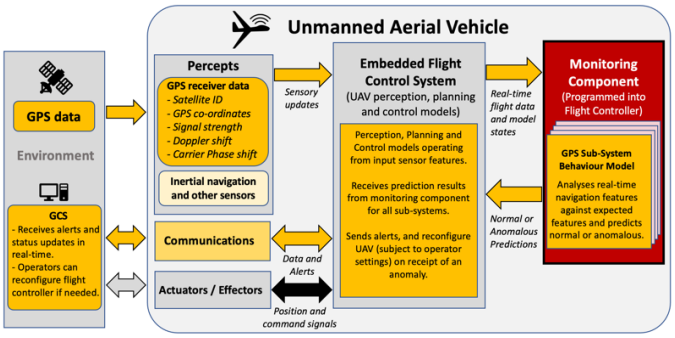

Fig. 8. UAV model for the proposed system detecting GPS intrusions.

The monitoring component produced from the UAV digital-twin contains a GPS-specific model. This analyses the behaviour of the flight controller and its modelled states, and compares this against its own expected behaviour model. Whenever there are large discrepancies, the monitoring component predicts the GPS system as anomalous and informs the flight controller.

If the flight controller was pre-configured by the operators to automatically counter anomalies, it shall automatically reconfigure the navigation models to ignore GPS and use inertial navigation inputs for deduced reckoning inputs until the anomaly is lifted. At all times during this process, the system state, alerts and configuration is sent to the operators in real-time using the system communications.

IV. Methodologies and System Algorithms

A. *Anomaly and Novelty Detection*

The digital-twin uses novelty detection models to discriminate between normal and anomalous behaviour. Anomalies are assumed to arise when cyber-intrusions occur, and the digital-twin is used to detect these. Anomaly detection models identify inputs that deviate significantly from normal data, which relies on having enough normal and anomalous data to learn from [23]. Using UAV data with

normal and anomalous flights, a model can be trained to recognise anomalies. The problem is that anomalies are rare and difficult to obtain in sufficient quantities for training [5]. There may also be unseen attacks that the model would fail on without past data.

These issues are overcome with novelty detection, which trains exclusively on normal (non-anomalous) data. Resources are not wasted on collecting examples for all possible anomalies, since only normal data is needed. After training the model can classify new data as normal or anomalous, provided there is a detectable difference between the distribution of features. Any deviation beyond a chosen threshold can be classified as anomalous. A cyber-intrusion is likely to make the system behave abnormally [3]. Thus, assuming an intrusion is reflected in the data, a novelty detection model can detect this, even for unseen attacks.

Model production involves two stages: training and inference. Historic UAV flight data is used to train normal-behaviour models. Once sufficiently tuned and tested, the model is deployed into the UAV and used for real-time inference on new data. The real-time data is pre-processed exactly the same as during training, and fed into the deployed model. Dependent on the variation between these new features and the expected features, the model will classify inputs, as illustrated inFig 9.

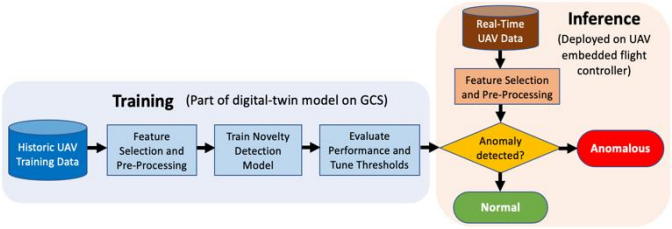

Fig. 9. Novelty detection process, with training and inference elements.

The training of the digital-twin takes place in the GCS, whilst real-time inference takes place on the UAV embedded flight controller using deployed versions of the models. Rather than training one large model the digital-twin contains models for each sub-system, which contain only the most relevant and insightful features from the data. This reduces complexity and the number of redundant features each model must process. For example, the GPS sub-system model would keep useful features for identifying spoofing or jamming, such as those relating to navigation, location, and positioning. Selection of key features is one of the roles played by experts in the digital-twin architecture.

B. Novelty Detection Algorithms

A wide variety of novelty detection algorithms exist, including distance-based, ensemble-based, statistical, domain-based, and reconstruction-based [5]. Due to the vast number available, only a subset of algorithms were considered during analysis and evaluation. These included One-Class Support Vector Machine (OC-SVM), Isolation Forest (IF), Local Outlier Factor (LOF), and various Deep Neural Network (DNN) encoder-decoder architectures. Each model was evaluated on real UAV data for detecting GPS spoofing attacks, which is a typical function expected from the digital-twin.

OC-SVM is a domain-based detection technique, which defines a boundary to separate normal instances from anomalous ones based on the training data. It is an extension of the Support Vector Machine (SVM) classifier to work with unlabelled data in an unsupervised learning context [24]. The SVM algorithm captures all data points from one class and separates them from another using a maximised margin. It does this by mapping the input data into a complex non-linear feature space. For linearly inseparable problems, this new feature space can provide projections that separate data instances effectively. Computation for this can be expensive, however kernel functions make this more efficient. A common choice is the Gaussian kernel, which provides a mapping into a higher-dimensional dot product space [25]:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)) \qquad (1)$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right) \qquad (2)$$

Where $\mathbf{x}_i$ and $\mathbf{x}_j$ are vectors with the i^{th} and j^{th} feature values of training data, Φ is a chosen mapping function, and σ is a free-parameter optimised for the problem.

After feature space mapping, the separation of the data classes is achieved through maximising a hyperplane of separation using convex quadratic programming techniques. OC-SVM is similar to the standard SVM model described, except it only uses one class representing normal data. After training, the model maps input data into the feature space, and predicts whether it is normal or anomalous based on its relative position.

The IF algorithm uses a large number of decision trees to classify instances as normal or anomalous based on anomaly scores computed for each instance. Anomalies tend to be easier to isolate, and therefore tend to have shorter paths lengths compared to normal instances [26]. The anomaly score for each instance is calculated by using the path lengths from the root to the terminating nodes. Those with shorter paths are more likely to be anomalies, whilst longer ones are more likely normal. The benefits of this algorithm include high-performance on high-dimensional datasets and its fast computation time for large datasets [26], both of which are important for UAV data.

LOF is a clustering-based algorithm that works by finding the k-nearest neighbours and identifying local outliers at each cluster. An anomaly score is determined for each instance, which is the ratio between the density of the instance and the average density of the nearest cluster [5]. This works well for detecting sparse outliers when trained on representative dense regions of normal data. However, such distance-based clustering does not scale well to high-dimensions due to the curse of dimensionality, and so feature reduction is often required.

A neural network is a machine learning model inspired by human biological neurons. Their form can be modified to create many different architectures, one of which is an encoder-decoder [27]. These architectures, known as autoencoders, employ reconstruction-based methods suitable for novelty detection. This works on the principle that anomalous data loses more information when reduced to a lower-dimensional space compared to normal data [5].

These models have equal numbers of input and output neurons, and one or more hidden layers with fewer neurons. Since the inner layers have less neurons than the outer-layers, the network reduces the data dimensionality as it passes through the network [5]. The model then reconstructs the original data as effectively as possible from the lower-dimensional latent representation, as shown in Fig 10.

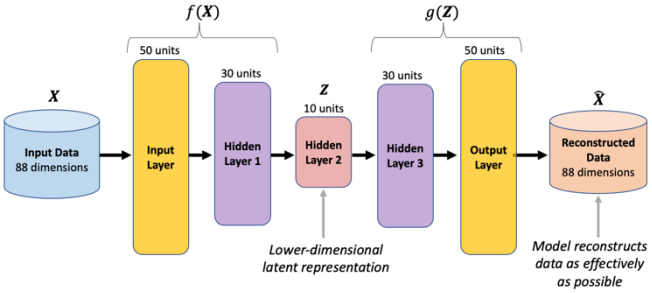

Fig. 10. Encoder-Decoder architecture representation.

The first component is the encoder (Equation 3), which compresses the input features to a lower-dimensional latent vector. The decoder component (Equation 4) follows this by reconstructing the latent vector back to the original dimensionality.

$$\boldsymbol{Z} = f(\boldsymbol{X}) \qquad (3)$$

$$\widehat{\boldsymbol{X}} = g(\boldsymbol{Z}) \qquad (4)$$

Where f is the encoder, g is the decoder, $\mathbf{Z}$ is the latent representation, $\boldsymbol{X}$ is the input features and $\widehat{\boldsymbol{X}}$ is the reconstructed features.

In general, any number of hidden layers with non-linearities can be used for the encoder and decoder to produce deeper networks with greater learning capacities [25]. The basic layers in Fig 10 are generally fully-connected (dense) by default, but may be modified appropriately to create more complex architectures including Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) autoencoders.

Since the latent representation is lower-dimensional than the original data, the model is forced to encode the data efficiently. The decoder reconstructs the original data from this lower-dimensional representation. The models are optimised through minimising the Mean Squared Error (MSE) reconstruction training loss (Equation 5).

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \widehat{\boldsymbol{x}}^{(i)})^2 \qquad (5)$$

Where $\mathbf{x}^{(i)}$ and $\hat{\mathbf{x}}^{(i)}$ are vectors consisting of the true features and reconstructed features for the i^{th} training example, and n is the total number of data instances.

This gives a vector of reconstruction losses for each feature averaged over the number of samples, n. The final training loss used for optimisation is the mean reconstruction error across all features, which is a scalar value. For novelty detection, anomalies are reconstructed poorly with large MSEs compared to normal data. Therefore, the reconstruction errors can be used to identify anomalies by setting a threshold above which instances are considered anomalous [5].

The basic autoencoder architecture can be enhanced by introducing noise into the encoder during training. This is known as a denoising autoencoder and can improve generalisation performance, since the network learns to reconstruct noise-free inputs [28]. A simple way to achieve this is to use dropout layers throughout the network to randomly disable different combinations of neurons during training iterations [29].

Three deep encoder-decoder architectures were considered in this work. The first was a Denoising Autoencoder, which takes point-based data as the input without consideration of time-dependency. The other architectures are time-sequencing models that learn and exploit patterns in data over time through using time-windowed input features, which included a convolutional autoencoder and Long Short-Term Memory (LSTM) autoencoder. Since UAV data consists of time-series data that changes over time, it is expected that a model that understands patterns over time would be more robust, as shown by [7] for ADS-B messages.

A CNN contains spatially local connections and applies kernels, which are patterns of weights replicated across many localised regions in an image [30]. Applying kernels to a set of input features is known as a convolution. Although mainly used for 2D data like images, 1D convolutions are effective for modelling one-dimensional features with sequential dependencies like UAV time-series data. In this regard, it can be formed into an autoencoder architecture using the same principles discussed above.

RNN architectures are networks that use cycles in the computation graph to maintain an internal state or memory that can exploit patterns across data over time [30]. An LSTM network is an RNN architecture with LSTM layers, which can be used to produce an autoencoder network using the same principles as the convolutional model.

A final consideration for autoencoder-based novelty detection models is the requirement to select a threshold (reconstruction error value) for identifying anomalies. Existing works like [7] select this by conducting k-folds cross-validation on training data and selecting the average value below which 95% of data is correctly classified as normal. A similar approach is used in this work, except this time a threshold factor is chosen, which is multiplied by the standard deviation and added to the mean of the reconstruction errors (Equation 6).

$$Threshold = \mu_{\mathcal{L}} + \alpha\sigma_{\mathcal{L}} \qquad (6)$$

Where $\mu_{\mathcal{L}}$ and $\sigma_{\mathcal{L}}$ are the mean and standard deviation of the reconstruction errors respectively, and α is the selected threshold factor for the specified model.

The threshold factor provides control of how likely a data instance is to be classified as anomalous relative to the training data. This is important to set appropriately, so that similar flights are not inadvertently classified as anomalous. This was validated during analysis using an allocated sub-set of the test data containing anomalous instances.

V. Analysis, Validation and Discussions

A. Overview of Analysis

For analysis GPS spoofing attacks were chosen to demonstrate the design performing novelty detection. This process would be similarly conducted for all other UAV sub-systems. The performance of six different novelty detection models are compared, including OC-SVM, IF, LOF, denoising autoencoder, and time-sequencing convolutional and LSTM autoencoders. The results are used to discuss the most suitable model for the digital-twin architecture.

The open-source UAV attack dataset is used [31], which contains a mixture of normal flights and GPS-spoofed flights for six UAV platforms. A similar process is followed relative to [11] and [12], who also perform novelty detection on this dataset. However, improvements are made, including more generalisable models across all datasets, and enhanced interpretability of predictions. These elements are both important for the digital-twin design.

The data is provided as down sampled UAV log data, representing normal behaviour and a small subset of spoofed instances (Table II). The authors provide no processed ground-truth labels for anomalies, however they provide the time and duration of each spoof. This was used to hand-label anomalies for model evaluation.

TABLE II. Data provided within the UAV Attack Dataset.

	Normal Flight Data Instances	GPS Spoofed Data Instances
Yuneec H480 Hexacopter	2971	44
PX4 Standard Plane	1406	46
Holybro S500 Quadcopter	3201	46
3DR IRIS+ Quadcopter	3858	44
PX4 Standard Tailsitter	1163	46
DeltaQuad VTOL	1420	50

Each UAV flight used a PX4 Autopilot and a Pixhawk 4 controller. The GCS used the QGroundControl software for UAV planning, data processing and control. All flights are provided as time-series in .csv format, with a single timestamp per row. Each row contains 88 UAV features, such as airspeed, altitude, pitch, heading, GPS co-ordinates, and climb-rate. All UAVs had similar flights, with minor variations between rotary and fixed-wing designs, as shown in Fig 11.

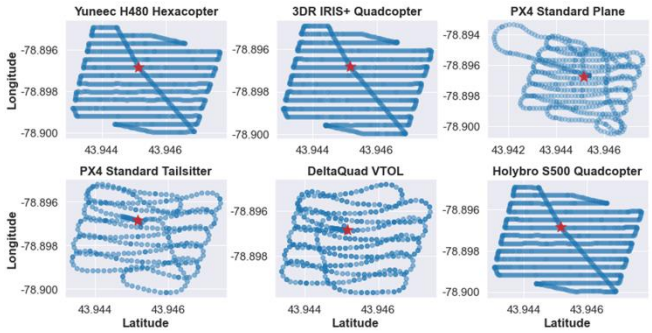

Fig. 11. GPS co-ordinates for normal flights from the UAV Attack Dataset.

The spoofed data is similar, except the GPS is spoofed at some random point, resulting in the UAV being fooled for thirty seconds. The purpose of the models is to identify when the UAV is being spoofed, which would happen in real-time for the proposed system using the deployed monitoring component from the digital-twin architecture.

B. Data Preprocessing

The data is high-dimensional with 88 UAV features. Many of these are redundant or null, and therefore cleaning and feature reduction was required. Many features had widely ranging scales, and so standardisation was applied to give each feature zero-mean and unit standard deviation (Equation 7).

$$x'_j = \frac{x_j - \mu_j}{\sigma_j} \qquad (7)$$

Where $\mathbf{x}'_j$ is the vector of transformed values for the j^{th} feature, $\mathbf{x}_j$ is the original j^{th} feature vector, μ_j is the feature mean sample, and σ_j is the feature standard deviation.

Furthermore, the OC-SVM, isolation forest and LOF models performed poorly with high-dimensions, and required dimensionality reduction. Therefore, Principal Component Analysis (PCA) was used with a suitable proportion of variance retained. The autoencoder architectures did not require this and performed well by default. For the OC-SVM, isolation forest, LOF and autoencoder models the preprocessing operations are summarised in Fig 12.

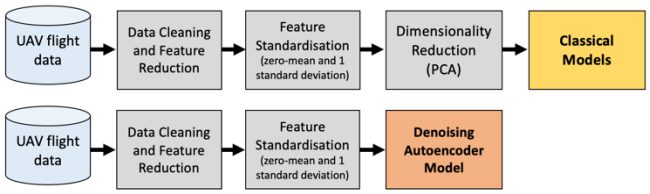

Fig. 12. Preprocessing pipelines various model types.

Fig. 13 shows the convolutional and LSTM autoencoders model time-sequences and past values, which adds preprocessing complexity. The data was sorted by time for each flight, and processed into time-sequencing sliding windows.

Fig. 13. Preprocessing pipeline for the Convolutional Autoencoder time-sequencing model.

All preprocessing techniques are summarised in Table III.

TABLE III. Preprocessing operations for each model.

Model	Standardisation Technique	Dimensionality Reduction	Time Sequence Processing
OC-SVM	Standard scaling	PCA, 85% variance	None
IF	Standard scaling	PCA, 90% variance	None
LOF	Standard scaling	PCA, 95% variance	None
Denoising Autoencoder	Standard scaling	None	None
LSTM Autoencoder	Standard scaling	None	Time-ordered sliding windows of length 6
Convolutional Autoencoder	Standard scaling	None	Time-ordered sliding windows of length 4

C. Model Production

All analysis was performed using Python open-source packages, with Pandas used for data loading and manipulation [32], and Scikit-Learn [33] for preprocessing and creation of the classical models. Keras was used for all autoencoder models [34].

Rather than optimising model hyper-parameters separately for each UAV, the analysis sought models that maximised performance across all platforms. This prioritised robust models that do not require bespoke tuning for each UAV and therefore provide better generalisation, which is important for the digital-twin. This was performed using hyper-parameter grid-searches across a wide range of potential combinations with k-folds cross-validation on the training and validation data. This gave the final model hyperparameters summarised in Table IV.

TABLE IV. MODEL FINAL HYPER-PARAMETERS.

Model	Library and Model used	Hyper-Parameters
OC-SVM	Scikit-learn OneClass SVM	***kernel**='rbf', **nu**=0.0005, **gamma**=0.0001*
IF	Scikit-learn IsolationForest	**contamination**=0, n_**estimators**=500, **max_features**=1.0, **bootstrap**=False, **max_samples**='auto'
LOF	Scikit-learn LocalOutlierFactor	n_**neighbours**=20, **algorithm**='auto', **leaf_size**=50, **novelty**=True
Denoising Autoencoder	Keras Sequential Dense Layers	**hidden layers**=[50, 3, 50], **activation**='selu', **dropout**=0.3, **loss**='MSE', **optimizer**='adam', **epochs**=80, **batch_size**=64, **threshold_factor**=30
LSTM Autoencoder	Keras Sequential LSTM Layers	**lstm_hidden_layers**=[64, 32, 4, 32, 64] **loss**='MSE', **optimizer**='adam', **epochs**=50, **batch_size**=128, **threshold_factor**=20
Convolutional Autoencoder	Keras Sequential Conv1D layers	**hidden layer filters**=[32, 16, 32], **hidden layer kernels**=[7, 7, 7], **stride**=2, **activaton**='relu', **dropout**=0.4, **loss**='MSE', **optimizer**='adam', **epochs**=50, **batch_size**=128, **threshold_factor**=20

Although each autoencoder was allocated a number of training epochs (Table IV), these figures represent upper-limits since model training was automatically stopped when the rate of loss improvement per epoch dropped below 0.2.

D. Metrics for Performance Evaluation

For anomalies, evaluating performance using accuracy is misleading, given the imbalanced nature of the data. Only 1% of the dataset contains anomalies, and always predicting normal yields 99% accuracy. Thus, better metrics are needed that consider True Positives (TPs), True Negatives (TNs), False Positives (FPs), and False Negatives (FNs). Good solutions include precision, recall and F1 score [25].

Precision is the proportion of instances correctly classified as anomalous out of all anomalous predictions (Equation 8). Recall is the proportion classified as anomalous out of the total anomalous instances, often referred to as True Positive Rate (TPR) or Detection Rate (DR) (Equation 9). F1 combines recall and precision, and is a good compromise of both (Equation 10).

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

$$Recall = \frac{TP}{TP+FN} \quad (9)$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (10)$$

High recall is important since it ensures the system does not let anomalies go unnoticed. Conversely, many false-positives could overwhelm the system and operators, and so precision must also be high. The balance between both is a compromise, and maximising one can impact the other. Thus, a model that maximises both could be considered best, as identified by the highest F1 score. Accuracy, precision, recall and F1 score were all calculated for model evaluation.

The threshold for classifying anomalies also impacts model performance. Therefore, the Receiver Operating Characteristic Area Under the Curve (ROC AUC) was also computed. This metric ranges from 0 to 1 and quantifies how performant the TPR is relative to the False Positive Rate (FPR) across all thresholds [25]. A score of 1 is best, however the results need careful consideration since the score does not consider specific thresholds, but only represents how well the model can perform if it is calibrated to the best threshold. In practice, a specified threshold or range of thresholds is needed for the model before flight, and therefore a model with the best ROC AUC could still perform poorly with an inappropriate threshold.

E. Results and Evaluation

With six UAVs in the dataset, and hence six unique anomaly detection problems, the overall average metric values were used to compare the model performance and suitability (Table V and Fig 14).

TABLE V. MEAN RESULTS FOR NOVELTY DETECTION MODELS ACROSS ALL SIX UAV DATASETS.

Model	Accuracy	Precision	Recall	F1	ROC AUC
LOF	0.7949	0.5612	0.7977	0.5543	0.9247
OC-SVM	0.9360	0.6263	0.6194	0.6197	0.7037
IF	0.9800	0.7918	0.8008	0.7896	0.9503
Denoising AE	0.9935	0.9504	0.9967	0.9638	1.0000
LSTM AE	0.9972	0.9906	0.9788	0.9833	1.0000
Conv AE	0.9975	0.9765	0.9950	0.9851	1.000

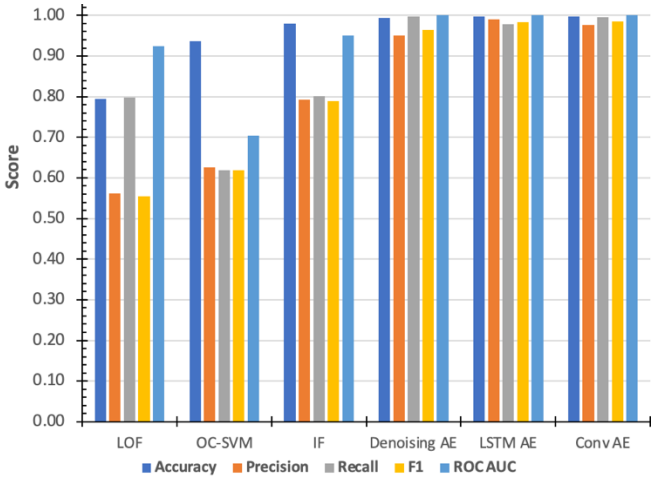

Fig. 14. Performance results across all UAV datasets.

In general, the classical models (OC-SVM, IF and LOF) performed poorly across multiple datasets using the same

hyper-parameter settings, as illustrated inFig 14. Each struggled to generalise to the high-dimensionality data, which made dimensionality reduction essential during pre-processing. This reflects the lower learning-capacities of these models in contrast to deep autoencoders, which performed well using all features as inputs. Although the classical models were simple and computationally efficient, the results suggest they might not perform well enough to be feasible without significant improvements or changes to the modelling process used in this work.

The results show the autoencoder-based models were superior, with the time-sequencing convolutional and LSTM autoencoder variants performing marginally better across all metrics. This reflects the higher model expressiveness from autoencoders on high-dimensional time-series datasets, in contrast to the classical models.

The time-sequence autoencoder models appear to be marginally better than the denoising autoencoder, which could suggest these benefitted from additional insights from time-sequences, rather than independent instances. Such modelling allows time-dependent aspects to be represented, such as trajectories and co-ordinate displacements resulting from velocity, heading and actuator position changes, making anomalies easier to spot. However, it should be highlighted that these conclusions cannot be drawn from this work alone due to the constrained and limited size of the UAV dataset considered, in conjunction with the marginal performance differences observed. It should also be highlighted that such time-dependent models have a tendency to overfit smaller training datasets, and therefore much larger and representative flight data would be needed to ensure they generalise reliably. Further experimentation with more complex, diverse and sufficiently large UAV time-series datasets is required to confirm these conclusions, which was beyond the scope of the scope of this work.

Further limitations of the autoencoder-based models include the increased complexity and longer training time, especially the LSTM. Fortunately, these models would not be trained during flight, but instead on the GCS before deployment to the UAV. Inference times were insignificant in comparison to other models during analysis, and so this is unlikely to be problematic. However, this would need validation through physical trials on UAV embedded devices using much larger datasets to ensure no real-time computational issues arise.

Based on the experimentation results, the UAV digital-twin would perform best for intrusion detection using an autoencoder-based model. In particular, the convolutional variant appears optimal from those evaluated, since it performed best overall and has the capacity to model time-dependent features. Despite this, the results were very similar and further experiments would be recommended on larger and more representative UAV datasets to analyse this in practice. Furthermore, in terms of computational efficiency and speed, the convolutional variant was much faster to train compared to the LSTM model, which further advocates the convolutional model for this application.

F. Further Autoencoder Architecture Benefits

In addition to enhanced performance across all metrics, the autoencoders bring valuable interpretability and explainability capabilities for anomalies and intrusions. The reconstruction losses can be calculated for each individual input feature, and these can highlight specific features that were difficult to reconstruct for the given inputs. During experimentation, the mean and standard deviation for feature reconstruction losses were calculated on the test data, with a sample of mean results, shown in Fig 15.

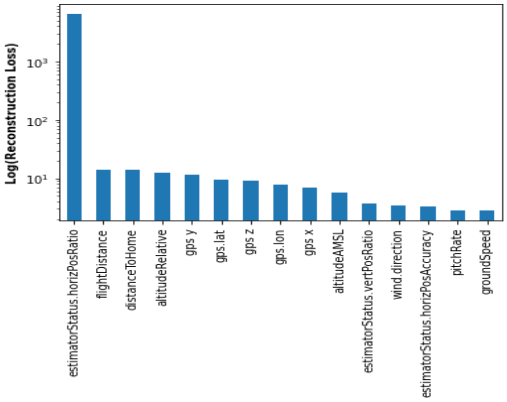

Fig. 15. Mean test feature reconstruction errors for top 15 features using the denoising autoencoder.

A high mean feature reconstruction error suggests that feature varies significantly to the expected UAV behaviour. Similarly, a high standard deviation suggests high uncertainty for a given feature. These results would be useful as feature importance scores to indicate to UAV operators in real-time why particular inputs were classified as anomalous.

Additionally, the autoencoder latent encodings provide insights into how similar new data is relative to expected data. Low-dimensional latent representations can be visualised to compare anomalous and normal data samples, as shown in Fig 16 for predictions on the 3DR IRIS+ quadcopter. This can help visually indicate how significant an anomaly is relative to normal behaviour, which could be valuable for both in-flight monitoring and post-flight analysis of faults, intrusions or anomalous events, thus extending the digital-twin capabilities further.

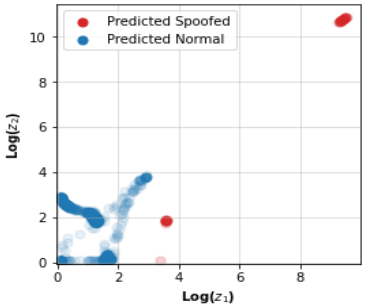

Fig. 16. Autoencoder latent encodings and predictions for test data on the 3DR IRIS+ quadcopter.

G. Final Discussions

From this analysis, the prospect of using a digital-twin for UAV cyber-intrusion detection is promising. As previously described, deployed versions of the models would be programmed into each UAV controller to act as a monitoring component, thus working in real-time across all sub-systems. This concept worked against GPS spoofing, where novelty detection models were effective against

sensor-based intrusions and anomalies resulting in deviations from normal behaviour. The classical novelty detection models struggled with the high-dimensional UAV data in this case, in contrast to deep autoencoders that performed significantly better, with exceptional results on the chosen example use-case.

The predictions demonstrated in these experiments would feed directly into the flight controller. If preconfigured to do so, the flight controller could automatically reconfigure or reset the system to mitigate any safety risks. Otherwise, the operators would be alerted through the real-time communications, thus allowing them to respond appropriately. The level to which a human remains in the loop in this case would depend on the safety context and mission nature.

To further validate the concepts of the proposed digital-twin design, these models could be evaluated on larger UAV datasets with a greater variety of sub-system cyber-intrusions, such as ADS-B, data-link communications, and inertial navigation sensor attacks. Furthermore, there is potential to develop this work by moving towards deep generative and probabilistic modelling techniques within the digital-twin. Architectures like Variational Autoencoders (VAEs) and variants thereof are effective for unsupervised novelty detection [35], along with Generative Adversarial Networks (GANs) [36] and deep autoregressive architectures [37]. Such architectures could provide more useful representations of UAV behaviour since they model the underlying distributions of the data, thus enabling generation of simulated flight data to complement the digital-twin architecture and produce more robust models. The challenge in this context is modelling long sequences of high-dimensional time-series data using generative models, rather than independently and identically distributed point data.

VI. Conclusions

This paper demonstrated the impact of digital twin architecture through the production of novelty detection models from historic flight data, which were demonstrated against GPS spoofing intrusions with good results. The digital-twin is instrumental in modelling the UAV using vast expanses of past data and inputs, whilst the deployed monitoring component on the UAV is responsible for putting these models to practical use through real-time inference of cyber-intrusions and anomalies to enhance security.

Further modelling techniques could have been investigated during the analysis. In particular, deep unsupervised learning architectures such as VAEs and GANs could be adapted to provide the benefits of generative models, such as deeper understanding of UAV behaviour and simulation of vast quantities of new data. However, the main priority of this work was not to evaluate the best anomaly detection technique for aviation-related data, but rather to demonstrate the feasibility of a digital twin architecture that applies data-driven algorithms to perform real-time detection against intrusions and cyber-threats. Therefore, investigating the feasibility and potential of such state-of-the-art models for UAV data could prove valuable for future research.

Additionally, future work could develop the concepts from this work into a physical prototype with a collection of behaviour models trained from historic flight data. These could be turned into novelty detection models for each major sub-system, as was performed for the GPS sub-system in this paper. Finally, these models could be tested during real flight trials and simulated attacks against the UAV, and the results used to validate the true potential of the proposed system.

Acknowledgment

This research was supported by the grants received from Innovation UK under the programme of Future Flight Challenge Phase 2: Strand 1, Development (Grant No# 75528- UAS Authentication Service (UASAS).

References

[1] J. Wurm *et al.*, “Introduction to Cyber-Physical System Security: A Cross-Layer Perspective,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 3, no. 3, pp. 215–227, Jul. 2017, doi: 10.1109/TMSCS.2016.2569446.

[2] M. L. Psiaki, T. E. Humphreys and B. Stauffer, "Attackers can spoof navigation signals without our knowledge. Here's how to fight back GPS lies," in *IEEE Spectrum*, vol. 53, no. 8, pp. 26-53, August 2016, doi: 10.1109/MSPEC.2016.7524168

[3] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, “Cyber Attack Vulnerabilities Analysis for Unmanned Aerial Vehicles,” Jun. 2012. doi: 10.2514/6.2012-2438.

[4] C. G. L. Krishna and R. Murphy, “A review on cybersecurity vulnerabilities for unmanned aerial vehicles,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017, pp. 194–199. doi: 10.1109/SSRR.2017.8088163.

[5] L. Basora, X. Olive, and T. Dubot, “Recent advances in anomaly detection methods applied to aviation,” *Aerospace*, vol. 6, no. 11. MDPI Multidisciplinary Digital Publishing Institute, 2019. doi: 10.3390/aerospace6110117.

[6] M. R. Manesh, J. Kenney, W. C. Hu, V. Kumar Devabhaktuni, and N. Kaabouch, “Detection of GPS Spoofing Attacks on Unmanned Aerial Systems,” 2019.

[7] E. Habler and A. Shabtai, “Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages,” *Computers and Security*, vol. 78, pp. 155–173, Sep. 2018, doi: 10.1016/j.cose.2018.07.004.

[8] V. Sadhu, S. Zonouz, and D. Pompili, *On-board Deep-learning-based Unmanned Aerial Vehicle Fault Cause Detection and Identification*. 2020. doi: 10.0/Linux-x86_64.

[9] X. Tan, S. Su, Z. Zuo, X. Guo, and X. Sun, “Intrusion detection of UAVs based on the deep belief network optimized by PSO,” *Sensors (Switzerland)*, vol. 19, no. 24, Dec. 2019, doi: 10.3390/s19245529.

[10] S. Khan, C. F. Liew, T. Yairi, and R. McWilliam, “Unsupervised anomaly detection in unmanned aerial vehicles,” *Applied Soft Computing Journal*, vol. 83, Oct. 2019, doi: 10.1016/j.asoc.2019.105650.

[11] J. Whelan, T. Sangarapillai, O. Minawi, A. Almehmadi, and K. El-Khatib, “Novelty-based Intrusion Detection of Sensor Attacks on Unmanned Aerial Vehicles,” in *Q2SWinet 2020 - Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Nov. 2020, pp. 23–28. doi: 10.1145/3416013.3426446.

[12] K. H. Park, E. Park, and H. K. Kim, “Unsupervised fault detection on unmanned aerial vehicles: Encoding and thresholding approach,” *Sensors*, vol. 21, no. 6, pp. 1–17, Mar. 2021, doi: 10.3390/s21062208.

[13] C. Andrea, S. Schmitt, and S. Squartini, “Real-World Anomaly Detection by using Digital Twin Systems and Weakly-Supervised Learning,” *arXiv*. arXiv, Nov. 12, 2020. doi: 10.1109/tii.2020.3019788.

[14] R. Snijders, P. Pileggi, J. Broekhuijsen, J. Verriet, M. Wiering, and K. Kok, “Machine Learning for Digital Twins to Predict Responsiveness of Cyber-Physical Energy Systems,” Apr. 2020. doi: 10.1109/MSCPES49613.2020.9133695.

[15] C. Hajiyev, H. Ersin Soken, and S. Yenal Vural, *State Estimation and Control for Low-cost Unmanned Aerial Vehicles*, 1st ed. Springer, 2015.

[16] E. Pastor, J. Lopez, and P. Royo, "A hardware/software architecture for UAV payload and mission control," 2006. doi: 10.1109/DASC.2006.313738.

[17] E. Dahlman and K. Lagrelius, "A Game of Drones: Cyber Security in UAVs," Stockholm, 2019.

[18] R. Altawy and A. M. Youssef, "Security, privacy, and safety aspects of civilian drones: A survey," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 2, 2017, doi: 10.1145/3001836.

[19] K. Hartmann and C. Steup, "The vulnerability of UAVs to cyber attacks - An approach to the risk assessment," 2013.

[20] E. Lee and S. Seshia, *Introduction to Embedded Systems*, 2nd ed. MIT Press, 2017.

[21] Q. Qi *et al.*, "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, Jan. 2021, doi: 10.1016/j.jmsy.2019.10.001.

[22] Y. Xu, Y. Sun, X. Liu, and Y. Zheng, "A Digital-Twin-Assisted Fault Diagnosis Using Deep Transfer Learning," *IEEE Access*, vol. 7, pp. 19990–19999, 2019, doi: 10.1109/ACCESS.2018.2890566.

[23] D. Tran, W. Ma, and D. Sharma, "Automated network feature weighting-based anomaly detection," in *IEEE International Conference on Intelligence and Security Informatics, 2008, IEEE ISI 2008*, 2008, pp. 162–166. doi: 10.1109/ISI.2008.4565047.

[24] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, and R. Holloway, "Support Vector Method for Novelty Detection," in *12th International Conference on Neural Information Processing Systems (NIPS'99)*, 1999, pp. 582–588.

[25] Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed. Packt Publishing, 2020.

[26] H. Chen, H. Ma, X. Chu, and D. Xue, "Anomaly detection and critical attributes identification for products with multiple operating conditions based on isolation forest," *Advanced Engineering Informatics*, vol. 46, Oct. 2020, doi: 10.1016/j.aei.2020.101139.

[27] A. Géron, *Hands-On Machine Learning With Scikit-Learn, Keras, And Tensorflow: Concepts, Tools, And Techniques*, 2nd ed. Sebastopol: O'Reilly Media, 2019.

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[30] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, 4th ed. Pearson, 2020.

[31] J. Whelan, T. Sangarapillai, and O. Minawi, "UAV Attack Dataset," Feb. 2020. doi: https://dx.doi.org/10.21227/00dg-0d12.

[32] W. Mckinney, "Data Structures for Statistical Computing in Python," 2010.

[33] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, [Online]. Available: http://scikit-learn.sourceforge.net.

[34] F. Chollet, "Keras," *https://keras.io*. 2015.

[35] C. Zhang, S. Li, H. Zhang, and Y. Chen, "VELC: A New Variational AutoEncoder Based Model for Time Series Anomaly Detection," Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.01702.

[36] P. Perera, R. Nallapati, and B. Xiang, "OCGAN: One-class novelty detection using gans with constrained latent representations," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2019, vol. 2019-June, pp. 2893–2901. doi: 10.1109/CVPR.2019.00301.

[37] T. Amarbayasgalan, V. H. Pham, N. Theera-Umpon, and K. H. Ryu, "Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error," *Symmetry*, vol. 12, no. 8, Aug. 2020, doi: 10.3390/SYM12081251.