

AI for Real-time Tolerance to Critical Flight Data Errors in Large Aircraft

Cynthia Koopman*

University of Malta, Msida MSD 2080, Malta

David Zammit-Mangion^{† ‡}

University of Malta, Msida MSD 2080, Malta

Cranfield University, Cranfield, Bedford, MK43 0AL, UK

The environment in the cockpit of large transport aircraft is highly complex due to an increasing amount of automation systems. This complexity can cause pilots to become less aware of how systems interact. It becomes a severe issue when sensor or data failures occur, as such failures can contribute to a situation in which it is difficult for a pilot to assess what actually is happening and, possibly, how to resolve the problem. This paper presents a method, based on artificial intelligence, for identifying incorrect critical flight control data in real-time. A novel combination of Reinforcement Learning and a denoising autoencoder is proposed to identify failures and to provide inputs to the aircraft's flight control and guidance systems, allowing for the correct manoeuvre to counter the failure and/or to avoid or recover from flight upsets. Tests in stall conditions with a partially blocked Pitot tube show that the proposed method results in successful detection and recovery. The performance of the system without an autoencoder is compared to highlight the significant advantages, how this relates to creating systems with AI to improve situational awareness for pilots and execute appropriate automatic manoeuvres to successfully counter the effect of sensor failures.

Nomenclature

A	=	advantage function
a_t	=	action at time t
c	=	weighting parameter
Δ_{target}	=	error to target
ϵ	=	deviation from old policy
f	=	encoder

*Ph.D Student, Institute of Aerospace Technologies, University of Malta, and AIAA Student Member.

[†]Director and Professor, Institute of Aerospace Technologies, University of Malta, and AIAA Senior Member.

[‡]Lecturer, Centre for Aeronautics, Cranfield University.

This work was presented as a conference paper at the AIAA AVIATION FORUM, 12-16 June 2023, San Diego, 4327

g	=	decoder
$\mathcal{L}$	=	loss function
π_θ	=	policy with parameters θ
$p_t(\theta)$	=	probability ratio between the old and new policy
P_{dyn}	=	dynamic pressure
P_s	=	static pressure
P_{tot}	=	total pressure
R	=	reward function
S	=	differential entropy
s_t	=	state at time t
θ	=	neural network parameters
V	=	value function
x	=	data sample
$\tilde{x}$	=	corrupted data sample

I. Introduction

Commercial aircraft today are complex machines that are highly automated with electronic systems that obtain information from a suite of sensors on board the aircraft. Aircraft automation systems have evolved over decades from relatively simple systems such as single channel auto pilots and navigators into complex machines that can completely manage the aircraft throughout the flight. This evolution, which has been influenced by technology, commercial constraints and certification requirements, has resulted in complex computational algorithms and processes that may not be fully understood by the pilot. In turn, this can, in certain operational conditions, lead to a disconnect between the automation and human actor. Such a disconnect can result in misunderstanding or mishandling of automation, possibly even to confusion on the part of the pilot; and this has the potential of causing significant incidents. It is also known that automation may result in an aircraft developing an undesirable state from which it is difficult or impossible to recover using traditional hand flying techniques [1]. The risk of such occurrences is exacerbated when sensor data is corrupted, such as in the case of a Pitot tube blockage.

In 1996, a Pitot tube blockage caused a Boeing 757 to crash off the Dominican Republic [2]. Although the automation on this aircraft may, to today's standards, not be considered complex, this accident nevertheless highlights the potential implications of data errors on automation and the risk of confusion on the part of the pilots when the automation acts in an unexpected manner. In the case of this accident, the crew were aware of a potential captain's airspeed indicator malfunction, but nevertheless engaged the central autopilot - which reads airspeed off the captain's system - on take-off

as per standard procedure. The airspeed indicator was over-reading, which led the autopilot to pitch up and reduce thrust in the climb, but also triggered the overspeed warning. The aircraft's speed dropped and approached the stall, triggering the stall warning simultaneously. The crew became confused, allowing the aircraft to stall and to remain in the stalled condition for about 80 seconds until it hit the sea.

A Pitot tube blockage caused another fatal accident 13 years later (2009), this time involving a more complex and highly automated aircraft, an Airbus A330 [3]. The blockage again resulted in incorrect airspeed data, of which the crew became aware. The incorrect airspeed data resulted in the autopilot disconnecting and the automation system entering 'alternate' law, which disables the stall protection system, and the auto-thrust became disconnected. The crew stalled the aircraft and, as in the case of the Boeing 757 accident, became confused and were unable to take the appropriate recovering action before the aircraft hit the sea.

More recently, the two Boeing 737 Max fatal accidents of 2018 [4] and 2019 [5] were triggered by incorrect angle of attack sensor data. These incorrect data caused the aircraft's MCAS system to push and trim the aircraft nose down on both occasions, overpowering the crew and resulting in a crash. Boeing 737 Max pilots were, at the time, not made sufficiently aware of the MCAS system and therefore could not be expected to understand what was happening at the time. This will have contributed to a greater disconnect between the pilots and automation; and the confusion on the part of the former.

There is an argument that if the crew could understand what is happening during an event, they may be in a position to take the correct action in a timely manner and avoid the situation becoming more serious. Indeed, the Boeing 757 can be flown safely to land with a Pitot tube blocked and had the crew of the accident aircraft known the cause of the problem, it could have been possible to avoid the accident. Likewise had the pilots of the Boeing 737 MAX accident flights understood that the MCAS system was repeatedly pushing the aircraft nose down, they could have pulled the circuit breaker off the stabilizer trim, thereby disabling the system. Indeed, this was done on the very flight prior to the 2018 accident, where the MCAS system was malfunctioning in a similar manner, and a disaster was successfully averted by the crew.

It is perhaps reasonable to expect the designer of an automation system to understand how it operates in detail. Even if considered challenging in certain situations, they might not fail to identify the source of a data fault and its implications in real time from the symptoms of automation behaviour. Leading them to potentially taking the appropriate and timely action to recover from the situation. However, it is not realistic to expect a pilot to have the same insight into the aircraft systems and it is clear that in such circumstances the flight crew's ability to act in an appropriate and timely manner will be compromised. A useful solution would be to have the knowledge of the system designer at hand and, in the absence of the designer in person on the flight deck, the use of Artificial Intelligence (AI) on the flight deck in the designer's stead constitutes an interesting proposition.

This paper presents, therefore, a method based on AI for detecting, identifying, and reconstructing incorrect critical

flight data as a basis to help the pilot understand what is happening in dynamic and failure conditions. Additionally, the proposed method provides instructions to direct the aircraft's flight control and guidance systems to conduct the correct manoeuvre. These flight control directions can counter a potential data failure to avoid or recover from flight upsets and loss of control, such as a stall.

In this context, a system that is capable of identifying erroneous data from a failed sensor and providing an effective estimate of the data pertaining to the failed sensor can be useful for the crew. Sensor readings often have distinct patterns and operate within a specific range of values. With an understanding of the sensor behavior over time, it is possible to identify a sensor failure (or data error) when it occurs. In addition, datasets in aircraft often need to be coherent. As simple examples, a drop in barometric altitude needs to be accompanied by a positive rate of climb, whereas a bank and turn to the right should result in a corresponding increase in compass reading. Pilots are trained to confirm displayed information in such ways. In this work, this concept of aircraft data needing to be coherent is exploited to monitor multiple variables used by multiple systems on board. All parameters will behave in a certain way during different flying maneuvers making them dependent on each other, and thus can provide important patterns for classifying a single parameter as defect. Since these patterns are highly complex, given the vast amount of possible combinations of multiple parameters, machine learning methods are used advantageously to correctly predict flight patterns and what indeed is happening when sensor or data failures occur.

Accordingly, an AI system that can estimate the correct state of the aircraft systems is created to provide insight to the flight crew and provide flight control directions to respond correctly despite faulty sensors or data errors. The estimate of the correct data of the aircraft sensor can either be used displayed to the pilots or used further for creating an alerting system, as explained later in this paper. The proposed approach is implemented by introducing a combination of an overcomplete denoising autoencoder and a Reinforcement Learning (RL) model. This combination learns how to fly the aircraft correctly and assess the systems for failures using the same neural network. Applying this combination of machine learning methods for this application is novel. It allows one neural network to be trained for two tasks; the reconstruction of the state (data) and estimating correct actions. This specific combination of tasks can be advantageous due to the similarity and thus create a deeper understanding and more meaningful features.

This paper is organized as follows. Section II discusses relevant previous work and highlights how the present approach differs from it. Section III presents the details of the architecture of the approach. Section IV describes the methodology of training the proposed architecture, including the simulation environment. Section V presents the results; those pertaining to the reconstruction of the sensor data in Section V.A, those relating to detection and alerting in Section V.B, and for directing the flight controls in stall recovery in Section V.C. These results are discussed in terms of the potential cost of combining multiple tasks in one neural network in Section VI and the conclusion is presented in Section VII.

II. Related Work

Failure detection systems can help pilots by increasing their awareness about the current state of the aircraft. Prior research employed data-driven methods where a neural network directly estimates the occurrence of one or more fault cases from the input sensors. Accardo et al. [6] and Zhao et al. [7] use this technique to detect air data system failures. Contrary to providing only an indication that an input is no longer reliable, this approach can offer more by also estimating the correct values of the sensors.

Depending on the type of failure, a time-prediction can also be made, and this acts as a health monitoring system on the sensor data. Ahmad et al. [8] and Berri et al. [9], use this technique to detect anomalies in individual sensors that are slowly degrading in performance (actuator and engine failures) in an attempt to predict a failure before it actually occurs. Approaches created for health monitoring can result in detecting very small changes over time, but these methods may not respond adequately to other sudden failure types. The failures considered in the current work typically occur due to causes other than gradual degradation and time prediction is therefore not considered feasible.

A different approach for the detection of failures involves the usage of autoencoders. This has been used for engine failure detection in UAVs [10, 11], for loss of control detection [12], and for cyber security related anomalies such as in a data bus in an avionics system [13]. These autoencoders are used for anomaly detection by learning the representation of the data when there is no failure in a latent space and then identify the occurrence of a failure when a significant deviation is detected. Anomaly detection autoencoders show promising results and constitute a practical approach when the dataset is highly imbalanced and does not contain enough instances of failures during network training. However, instead of merely signaling the unreliability of an input, the suggested method goes further by also estimating the accurate values of the sensors.

Reconstruction of the data errors during flight has already been studied. Such reconstructions can give pilots an indication of the current situation and can be used by systems that are dependent on the affected data to minimize propagating failures. With data-driven approaches, this has been done by a classical supervision approach and not using autoencoders. A classical supervision approach directly estimates a selected sensor parameter, such as airspeed, based on other sensor parameters that are assumed to be correct [14] [15]. Using neural networks, these reconstructions can be very accurate and can therefore be used for the detection of errors using a threshold for the allowable difference between the corrupted input and reconstructed input. The referenced studies show that there is enough data on board the aircraft during flight to estimate air system data accurately. However, for the failure types that are considered in the current approach it cannot be predicted which sensor will malfunction beforehand. The proposed approach therefore reconstructs all inputs and aims to provide a more holistic approach.

An aspect that is not sufficiently discussed in previous works is the detection and reconstruction of the sensor data during upset conditions. Also, the use of RL for directing flight controls outside the normal flight envelope has only been addressed by a few studies. In [16], an RL approach is used for stall recovery and another neural network is used to

stabilize the aircraft, which creates acceptable recovery strategies at various altitudes. In [17], a two-stage RL learning approach is used where first weight and wind are randomized and then fixed in the next learning phase. The work in [18] addresses wind to test the robustness of using RL for stall recovery and concludes its advantageous response compared to PID controllers. The results of these works show that the RL approach is capable of performing stall recovery in a variety of combinations of altitude, weight and wind. However, these RL approaches depend heavily on the correct operation of air data systems that provide the altitude and airspeed information. If these data are not correct, the RL systems are not able to provide the same performance. This inability can introduce challenges to the crew as they will have to detect when these recovery systems no longer operate correctly. This aspect can significantly contribute to the effects of loss of control.

This work proposes an approach that provides the crew with an alert when certain flight data is no longer valid, with sufficiently accurate estimates of the valid data (reconstruction) and outputs to direct the flight controls. The proposed system is designed with the aim to decrease instances as well as the consequences of loss of control. Consequently, this paper presents an architecture that, similar to previous work, compares the input with the reconstruction using a threshold, but further performs the reconstruction using a denoising autoencoder. The autoencoder learns the reconstruction of all input sensor data and when air data systems fail it is able to reconstruct the affected parameters. With this approach, there is neither need for prior knowledge on which data is affected nor are efforts made for feature selection, as the autoencoder is trained to reconstruct all sensor data inputs. Moreover, the denoising autoencoder approach was chosen because it can be strategically combined with an RL approach. This combination goes further than previous studies by providing fault detection, sensor reconstruction as well as the ability to direct the flight controls. All these tasks are performed using a single neural network, with significantly shallower neural connections compared to the deep networks used in previous work.

III. RL and Autoencoder Architecture

The proposed architecture combines an on-policy RL approach with an overcomplete denoising autoencoder as an auxiliary task in the same neural network. This architecture is summarized in Fig. 1, where the solid black lined objects form the system that is used after training, the dotted objects are used only for training, the red parts are for optimization, and the blue objects create a system that provides alerts for each state variable in real-time.

The RL component of this system provides the actions to direct the flight controls towards the target airspeed and altitude. This is implemented by using the Proximal Policy Optimization (PPO) algorithm [19]. PPO uses an actor-critic architecture where the actor uses a neural network to plan its actions based on the recommendation of the critic. The critic uses a separate network in the proposed architecture to predict the future reward, given the current policy. The critic does not take the manipulated state as input and thus does not need to perform any reconstruction of the state internally. This allows the critic to develop features that are beyond the reconstruction and improve the value

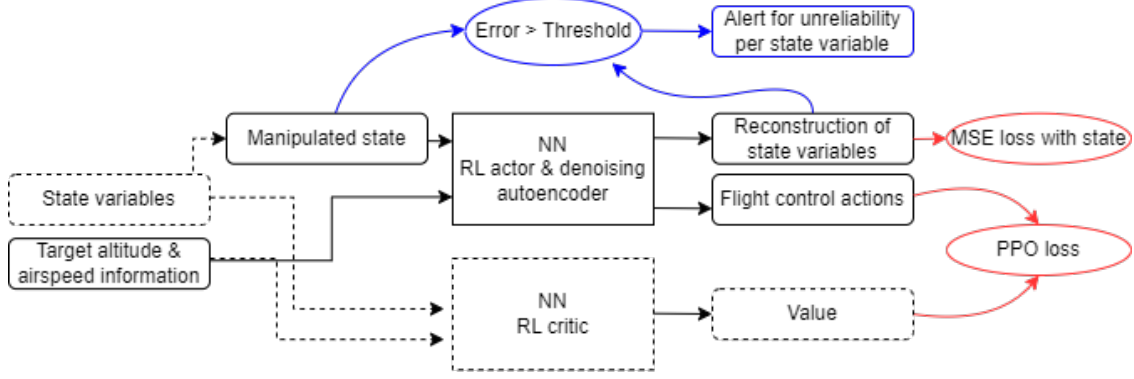


Fig. 1 System architecture for combining an on-policy RL agent with a denoising autoencoder.

prediction for the actor. Therefore, parameters are not shared between the actor and critic networks. The actor takes the manipulated state as input and is required to output actions that control the aircraft as if it is in nominal conditions. For optimization of the RL task, the PPO variant with a clipped objective is used, following the work by Schulman et al. [19] for continuous control tasks. This uses a probability ratio between the old and new policy $p_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$, to stabilize learning by limiting deviation from the old policy. The objective function for PPO follows Eq. 1:

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \min\left(p_t(\theta)A^{\pi_{\theta_k}}(s_t, a_t), \text{clip}(p_t(\theta), 1 - \epsilon, 1 + \epsilon)A^{\pi_{\theta_k}}(s_t, a_t)\right) \quad (1)$$

where ϵ is a hyperparameter that defines the amount by which the new policy is allowed to deviate from the old and $A^{\pi_{\theta_k}}(s_t, a_t)$ is the estimated advantage function based on the current value function $V_{\phi}(s_t)$. The critic network is optimized by the mean-squared error loss between the predicted value and the calculated advantages.

In addition to the RL task, an auxiliary task is learned in the form of a denoising autoencoder. This is implemented by adding an additional prediction head to the neural network of the actor with the same number of output units as the input state variables. Denoising autoencoders minimize the loss function $\mathcal{L}^{DAE}(x, g(f(\tilde{x})))$, where x is the data sample, $\tilde{x}$ the corrupted data sample, and f and g the encoder and decoder, respectively, created by the neural network. This task is learned by optimizing for the mean-squared-error loss between the correct input and the reconstruction. After training, the reconstructions can be compared to the erroneous input to generate an alert for each data input. The alerts are based on a detection using a threshold which is determined post-hoc.

The optimization of the actor, critic and autoencoder are all trained simultaneously during several training iterations. The complete optimization objective for the actor network for both tasks is summarized in Eq. 2:

$$\mathcal{L}_t^{CLIP+DAE+S} = \hat{\mathbb{E}}_t[\mathcal{L}_t^{CLIP}(\theta) + c_1 \mathcal{L}_t^{DAE}(\theta) - c_2 S[\pi_{\theta}](s_t)] \quad (2)$$

where $\mathcal{L}_t^{CLIP}(\theta)$ is the clipped PPO loss [19] which ensures that the policy does not deviate too far from the old policy,

$L_t^{DEA}(\theta)$ is the loss from the auxiliary denoising autoencoder task, and $S[\pi_\theta](s_t)$ is the entropy term. To ensure the main task is learned primarily, the hyperparameters c_1 and c_2 are used to balance the influence of the auxiliary loss and entropy term.

The architecture of the neural network of the actor is shown in Fig. 2. The actor and critic network only differ by output layer, where the critic network has only one output unit corresponding to the estimation of the value function. The network has an input layer that expands to 256 neurons into a Long Short Term Memory (LSTM) layer that provides time-dependent features, and is followed by two hidden layers each of 256 neurons. The output layer has two heads, corresponding to the actions and reconstructions of the input. The autoencoder task is implemented as an overcomplete one, where the input dimension is smaller than the internal dimensions. In classical autoencoders this can become problematic because the larger network has the ability to converge to an identity function where it is just copying input to output. In the proposed architecture, however, this limitation is mitigated and regularized by having the denoising task as well as the RL task for the same neurons.

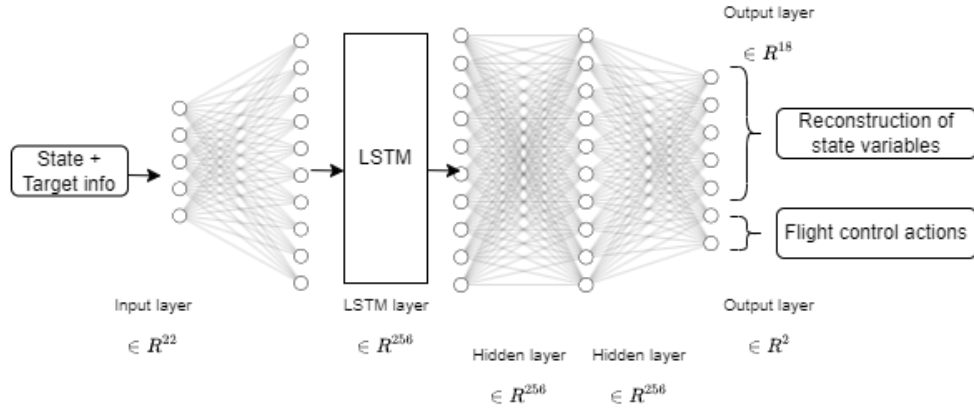


Fig. 2 Neural network architecture of the actor network.

IV. Training Set-Up

The aircraft and its surrounding atmosphere are simulated using the simulation software JSBSim [20] version v1.1.9. An A320 aircraft model is used for the implementation. This model is the native A320 model included in JSBSim version v1.1.9. JSBSim is an open-source software and more details can be found in the reference manual [21].

This simulation provides 6 degrees of freedom and can be used without visualizations and in a vectorized setting where multiple flight simulations can be used in parallel. These aspects significantly improve computation time and allow for the implementation of an on-policy RL algorithm.

The training space for RL is kept very large so that learning is enforced at different simulated operating conditions. The starting altitude is varied between 15,000 and 35,000 feet, and the starting true airspeed between 420 and 770 feet

per second (249 knots and 456 knots, respectively).

The reward for the RL agent was defined by the distance between the current and target combination of altitude and airspeed. The current distance to each target was first normalized between 0 and 1. This was done by forming a reward function with a sigmoid such that reward, for example, for altitude, started when within 500 ft of the target, achieved half of the possible reward (0.5) when within 250 ft of the target and a full reward (1.0) when on target. For airspeed, the reward started at 100 feet per second off target value and was half at 50 feet per second deviation.

This reward function is summarized in Eq 3:

$$R(s_t) = \frac{1}{N} \sum \text{sigmoid}(\Delta_{\text{target}}(s_t)) \quad (3)$$

where N is the number of targets that are included in the reward function and the sigmoid function is $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. The sigmoid function has the effect that far from the target large improvement can be made by the RL agent when moving in the correct direction, and close to the target the agent has to be more precise to gain more reward. The total reward was then a mean of the two normalized target rewards with equal weights.

The actions (outputs) of the RL agent were limited to the control of the elevator deflection and thrust setting. For simplicity, and due to simulation limitations, turns were not considered. Therefore, stalls that were performed were considered mundane in that they did not result in lateral dynamics such as a wing drop.

The state space (inputs) of the RL agent contains information that would be only available during flight. These inputs include both sensor/system readings and calculated information about the targets. The sensor/system readings that were included are barometric altitude, vertical speed, angle of attack, pitch angle, indicated airspeed, true airspeed, mach, temperature, speed of sound, static pressure, dynamic pressure, total pressure, current elevator and thrust position, engine speed (N1), and pitch trim setting. Using initially set target information, the following inputs were calculated: the set targets, total change in airspeed and altitude requested, and current distance to targets. Different readings of airspeed as well as components used for their calculation such as speed of sound and the pressures were included to provide the RL agent with more information when failures would occur. With these variables, the RL agent would be able to understand which inputs are affected by a Pitot tube blockage and which are not.

A Pitot static system was modeled to allow for failures during training. This system was implemented on top of JSBSim and used the pressure measurements provided by the said simulation. The pressure measurements available in JSBSim were total pressure P_{tot} and dynamic pressure P_{dyn} ; and these were then used to establish the static pressure P_s , which is the difference as $P_{\text{tot}} = P_{\text{dyn}} + P_s$. The Pitot static system comprised of a Pitot tube, a drain hole, and a static port. The relations between these components and airspeed and altitude measurements was created via standard pressure relationships which are also used by Pitot static system manufacturers [22]. In this work, only the Pitot tube was partially blocked. Blocking the Pitot tube restricted airflow in the tube while the air still in the tube could escape via

the drain, which corrupts the measurement of dynamic pressure. This has the effect of the airspeed information no longer being reliable, and when fully blocked will fall to zero. The resulting blockage affected the following parameters used by the agent: true and indicated airspeed, mach number, total pressure, dynamic pressure and as a consequence the calculated difference between current airspeed and target airspeed.

The Pitot tube blockage was normalised so that a partial blockage was represented by a continuous variable between 0 and 1. This facilitated the training of the neural network and the learning of the denoising autoencoder instead of an anomaly detection autoencoder. Anomaly autoencoders are typically trained on data that does not contain data samples with failures. They learn the structure of nominal conditions to create an internal representation in latent space. When they encounter states with failures they are then able to flag them as malicious, but this is not explicitly learned. In contrast, the RL agent requires to see a variety of sensor failures in order to extract the best features. An anomaly detection autoencoder would therefore not be able to train alongside an RL agent as the data is highly imbalanced for the anomaly detection task; whereas, for a denoising autoencoder, the data distribution is not only appropriate, but the different levels of severity in the Pitot tube blockages can also render learning features for reconstruction easier. These learned features are highly related to the RL task and can make learning overall more meaningful. Section VI addresses this concept further and includes a comparison of performance with and without the reconstruction task.

The final RL agent with auxiliary task was trained in the large training space while experiencing various combinations of partially Pitot tube blockages. During training, episodes were created by letting the agent interact with the environment for 1200 steps. These 1200 steps are taken at a frequency of 10Hz and thus translate to 120 seconds. Such episodes were collected for 256 parallel environments every training iteration. After each training iteration, the neural networks were optimized using the collected data comprised of episodes. Approximately 150 training iterations were needed for convergence.

V. Results

The results of the experiments evaluating the proposed architecture are presented in Section V.A for the reconstruction task; in Section V.B for the alerting task; and in Section V.C for directing the flight controls. It is important to note that in these experiments the same RL agent was evaluated and, specifically, these were not separately trained systems. The first two tasks were evaluated using episodes sampled from the same sampling space as used during training. The final experiments involved stall recovery episodes. These stall scenarios served as a testing set as well as conditions where pilots would not be able to successfully resolve the situation.

A. Reconstruction

The denoising autoencoder output units were evaluated by the Root-Mean-Squared-Error (RMSE) between the actual state and the reconstructed state. Figure. 3a shows the results per Pitot tube blockage level for the mean of all

variables. Here, the RMSE stays relatively low (numerically below 30) for blockages larger than 0.1, and only increases when the Pitot tube becomes completely blocked. However, this RMSE value is not very informative as it does not scale or weight each parameter. It therefore can only give an overall numerical trend (where the altitude effectively has a higher weighting than airspeed as the error values are higher). Figure. 3b shows the RMSE for each variable separately (only for those that were affected by the Pitot blockage). The most informative reconstruction for the crew during flight is for the airspeed, where our system achieves a mean RMSE that is always below 6 feet per second (3.6 knots), which is acceptable. When the blockage level goes below 0.5, the estimation of the difference in true airspeed to the target (delta TAS) starts to increase and deviate from the true airspeed estimation, reaching a maximum at full blockage with a RMSE of 8 feet per second (4.7 knots). The RMSE of the reconstruction of the total pressure stays below 0.2 in. Hg and the RMSE of the dynamic pressure below 0.08 in. Hg. These results show that the sensor failures caused by the partial Pitot blockage can be reconstructed with low error levels. The reconstruction of the sensor data by the denoising autoencoder is therefore considered successful and accurate enough for detecting (and alerting) in presence of a Pitot tube blockage.

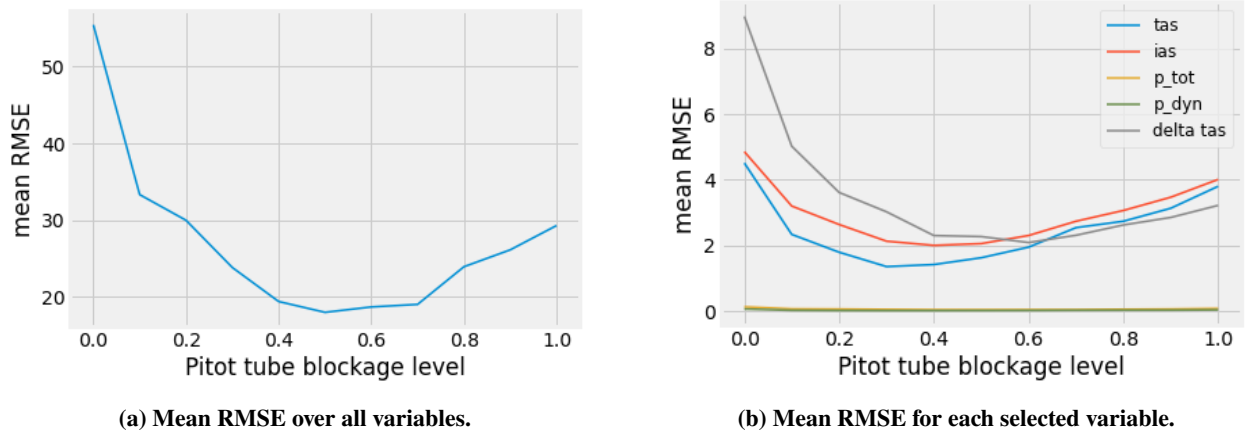


Fig. 3 Mean RMSE over 100 randomly sampled episodes for each Pitot tube blockage level, where 0 is a fully blocked Pitot tube and 1 is no blockage. Results shown for all variables and selected variables. True airspeed (TAS) and indicated airspeed (IAS) are in feet per second, total pressure and dynamic pressure in in. Hg.

B. Detection and Alerting

Using the reconstruction output of the denoising autoencoder, a threshold can be used to determine if the current input is valid or otherwise. This threshold was determined by evaluating on the accuracy, recall, precision and time prediction error for several different threshold settings, defined as a percentage of the range in which each variable occurred in the training space. For true airspeed, for example, 1% corresponds to 4.0 feet per second. The results are summarized in Fig. 4a, where the results are averaged over 100 randomly sampled episodes per threshold percentage setting. Less severe blockages (level > 0.5) were much harder to detect for the network as these result only in a slight deviation in the parameter. To ensure that the threshold is accurate for these more difficult blockages, the blockages

during the sampled episodes in Fig. 4a were sampled between 0.5 and 0.99 (not 0 and 1). The results show that having a threshold of only 1% has already an accuracy that is larger than 0.9 with a low average time prediction of 2.5 seconds.

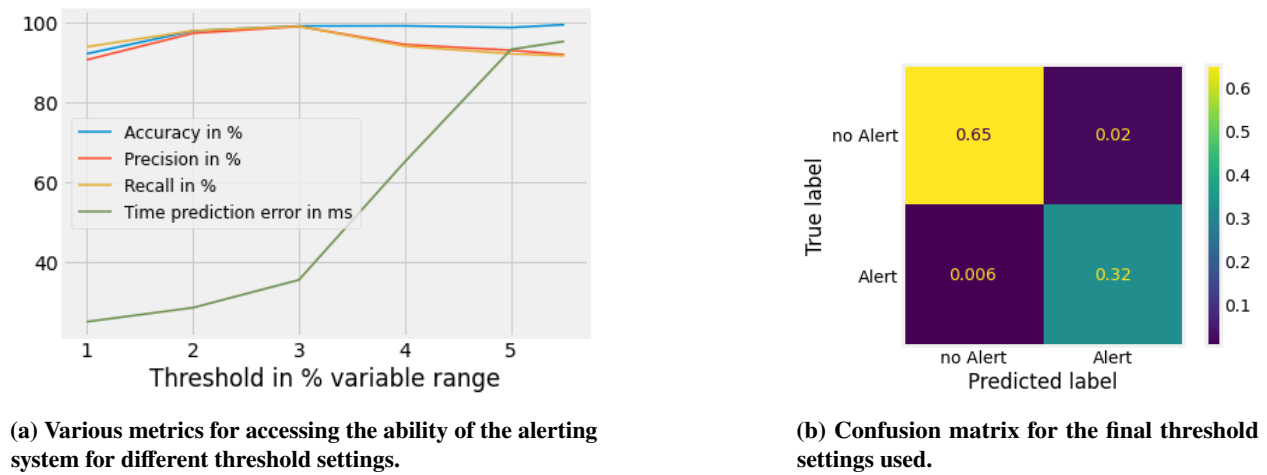


Fig. 4 Metrics and confusion matrix for the detection and alerting system assessing the difference between input and reconstructed variables. All results are over 100 episodes sampled from the training space.

By observing the results for such tests focusing on individual variables, it was found that only specific variables improved significantly with a higher threshold percentage than 1%. Consequently, this custom threshold setting of 1% was chosen for all variables except for static pressure (2%), trim setting (2%), elevator position (3%), N1 to (3%) and, thrust (4%). Increasing the threshold for these variables has the consequence of increased fault detection time, but was done to reduce the false positives and false negatives significantly. Overall, these settings resulted in a final performance of 0.98 accuracy, 0.98 recall, 0.97 precision, and an average time prediction error of 1.4 seconds. The corresponding confusion matrix is shown in Fig. 4b, which is for 100 sampled episodes of the training space with Pitot tube blockages between 0 and 1. This figure shows a 2% false positive rate and a 0.6% false negative rate. Figure 5 shows an example episode where a blockage is simulated after 600 episode steps. The green dotted line shows when an alert is activated for the specific sensor input. A small delay of a few seconds is visible for the total pressure, but for the other variables the blockage is detected immediately. The actual altitude and airspeed show that the RL agent is internally able to accurately reconstruct the input and is therefore able to approach the intended targets.

C. Directing the aircraft flight control system

The RL component of the proposed architecture provides actions that can be used to direct the flight control to the provided targets. This can be used either as an aid (for guidance/flight direction) for pilots or for (temporary) autonomous control. To test this ability, stall recovery scenarios were created. This was done by trimming the aircraft close to the stall speed, inducing a stall, and the RL agent taking over the controls as soon as the stall occurs. Stall recovery was chosen to test the RL actions because this is a scenario where a sensor fault such as a Pitot tube blockage

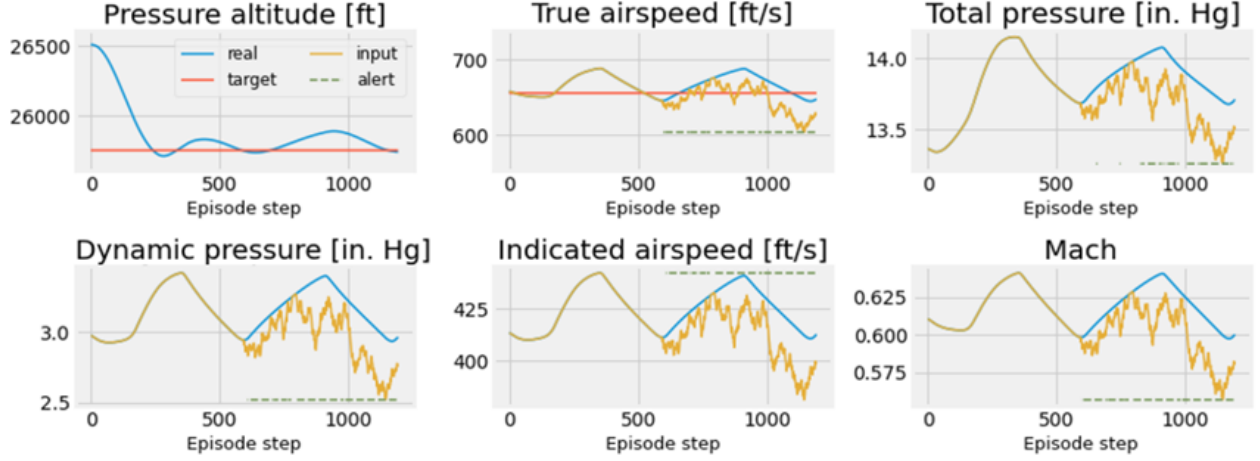
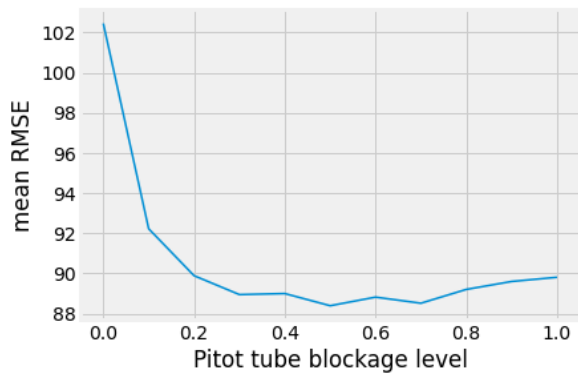
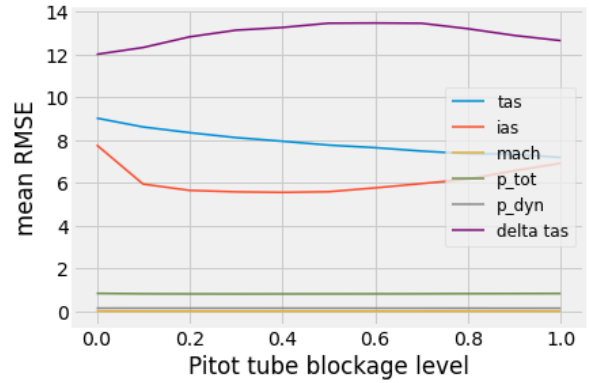


Fig. 5 Selected variables for an example episode where airspeed is maintained and altitude decreased. The Pitot tube blockages occur after 600 episode steps which are in 10Hz. The legends for all the figures are as shown for the pressure altitude.

is known to lead to stall and to loss of control. Nine different stall scenarios were created between 1,500 and 35,000 feet, which includes a large region below the limit of the training space (above 15,000 feet). This was done to test the recovery behavior outside the training space and test at low altitudes where stalls can be more dangerous. The target of the scenarios was arbitrarily to return to the original (pre-stall) altitude and increase speed by 100 feet per second (60 knots). Figure 6 shows the results for the RMSE of the reconstruction for all stall scenarios for different Pitot tube blockage levels. Overall, the RMSE slightly increased compared to that in the training space (Fig. 3a). Nevertheless, it remained low for both true and indicated airspeed; below 10 feet per second (6 knots). The difference of true airspeed to the target airspeed (delta TAS) is between 12 and 14 feet per second for all blockages and does not follow the same trend as previously seen in Fig. 3b. The estimates of the pressures remained similar to Fig 3b. For all stall test scenarios, the alerting was found to be immediate for all tests, with an average of 0.0246 seconds and a maximum error of 0.3 seconds. This is significantly lower than the average for episodes of the entire training space in the previous section, which could be because in stall a more accurate reconstruction performance is needed by the neural network to be able to succeed. The average accuracy of the alerts remained high at 0.93. Figure 7 shows three different examples of stall recovery whilst experiencing Pitot blockages. The first two examples have slowly decreasing or increasing blockages. The third example has a completely random blockage every episode step. Although this third example is perhaps not realistic, it is nevertheless noteworthy to assess how the AI system reacts to such. In each example, the agent was able to direct the flight controls to recover from the stall and return to original altitude, and a safe (low) angle of attack. As the RL agent was not trained specifically for stall recovery during training, these results show that the agent successfully learned useful features during training that allowed it to perform tasks that are considered far more demanding. These aspects also show that the RL agent is able to do this while reconstructing the input sensor readings and providing correct alerts to pilots.

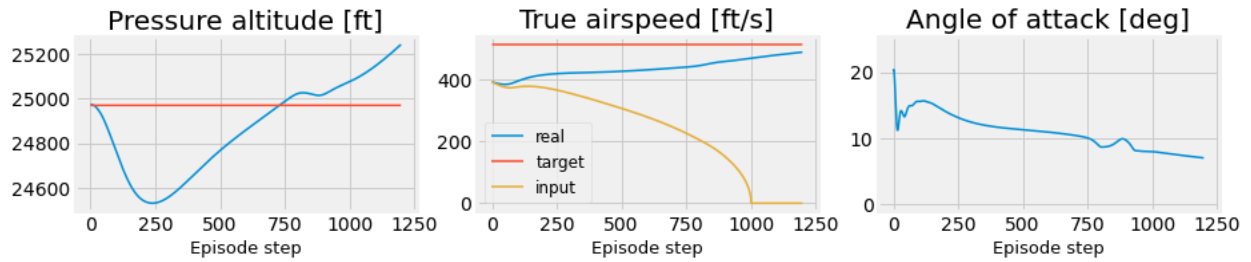


(a) Mean RMSE over all variables.

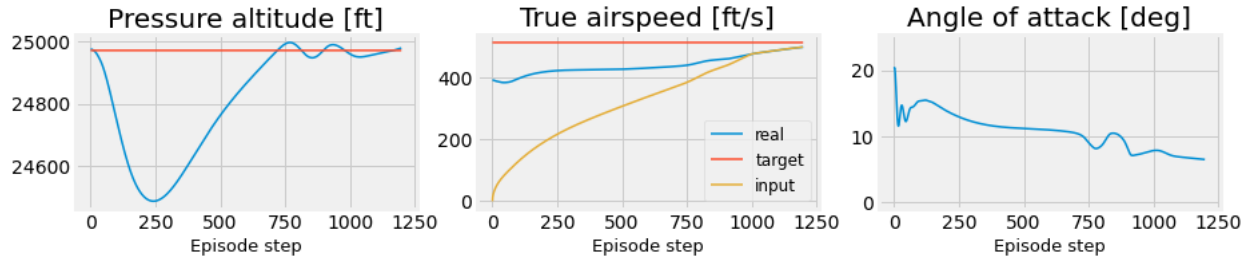


(b) Mean RMSE for each selected variable.

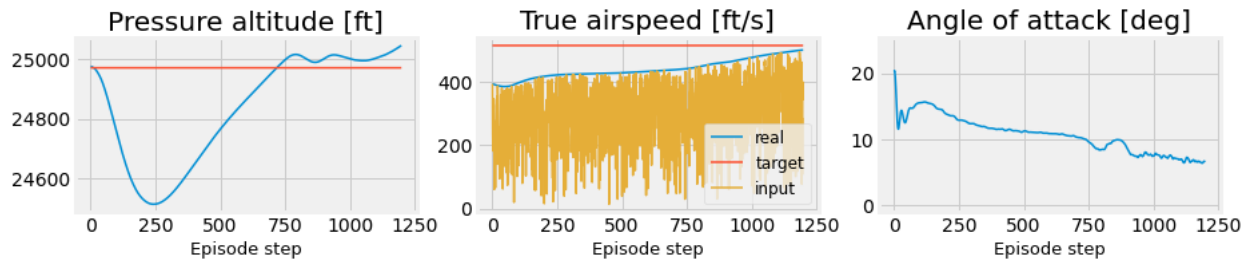
Fig. 6 Mean RMSE over 9 stall scenarios for each Pitot tube blockage level, where 0 is a fully blocked Pitot tube and 1 is no blockage. Results shown for all variables and selected variables. True airspeed (TAS) and indicated airspeed (IAS) are in feet per second, total pressure and dynamic pressure in in. Hg.



(a) Decreasing Pitot tube blockage.



(b) Increasing Pitot tube blockage.



(c) Random Pitot tube blockage.

Fig. 7 Selected variables for stall recovery examples for three different types of Pitot tube blockages. Episode steps are of 10Hz.

VI. The cost of the additional reconstruction task

In this paper, learning through the use of an RL model was combined with a denoising autoencoder model using the same neural network. A posing question is 'How much, if any, influence does this additional autoencoder task have on the RL model?'. To investigate this, a comparison was made between an RL model without the autoencoder task and the model trained in this paper.

It is not straightforward to presume the effect of including an additional task to a neural network. Ideally, the performance of the additional task does not have a large negative effect on the primary objective (the RL task). However, adding tasks to the primary objective can deviate attention, due to dissimilarity, and could therefore potentially degrade performance. If this degradation is small, the information gained through the additional task could be deemed worth while. However, if a task is too dissimilar to the primary task, it can significantly harm the learning of the RL task. For example, the current RL task could have had an additional task that directly outputs binary estimates whether an input is faulty or not, instead of the proposed autoencoder task. However, from our initial exploration, we observed that employing the former approach had a discernible impact on the learning process of the RL task. The reason behind the binary classifier not performing well may be due to the final layers of the network being forced to still provide information on which input was incorrect, instead of reconstructing in early layers and extracting more meaningful features deeper into the network. This shows the sensitivity and difficulty of adding a task.

Another possible outcome of comparing the proposed method with and without autoencoder task, is that the performance of both models is comparable. If this happens, it could potentially mean that the RL network was already reconstructing the input internally, and therefore the addition of the autoencoder task would extract the required information from the network. However, due to the inherent randomness of RL training and the complexities in neural networks, the actual representation and processing of information may differ from this claim.

The final possible outcome is that the proposed method has superior performance, which would strongly indicate that the autoencoder task actually influences the performance of the RL task in a positive manner. This described situation is observable for the proposed method. Figure 8 compares the mean reward obtained over several different Pitot tube blockage settings for the RL agent trained with the denoising autoencoder task (Fig.8a) and the RL agent trained without this task (Fig. 8b). A significant increase in performance with the denoising autoencoder can be observed. The RL agent with the autoencoder is not only able to gain higher reward, but also gains this across all levels of Pitot tube blockages. This difference clearly indicates that optimizing for the denoising autoencoder changes the internal connections of the neural network in a way that is extremely useful for the RL task. By being forced to learn this denoising task, the RL agent is effectively able to approach the RL task as if no data error occurs. Only for severe blockages (<0.2) the performance drops as too much information is lost. However, this denoising autoencoder is not only useful for the RL task and gives some insight into the neural network, but it also outputs information that is useful. The denoising task provides an accurate reconstruction of sensor data, which also allows for an accurate alerting system. This additional

information can be used to increase the situational awareness of pilots in case of data errors and/or provide insight into the actions of the RL agent when it takes control of the aircraft.

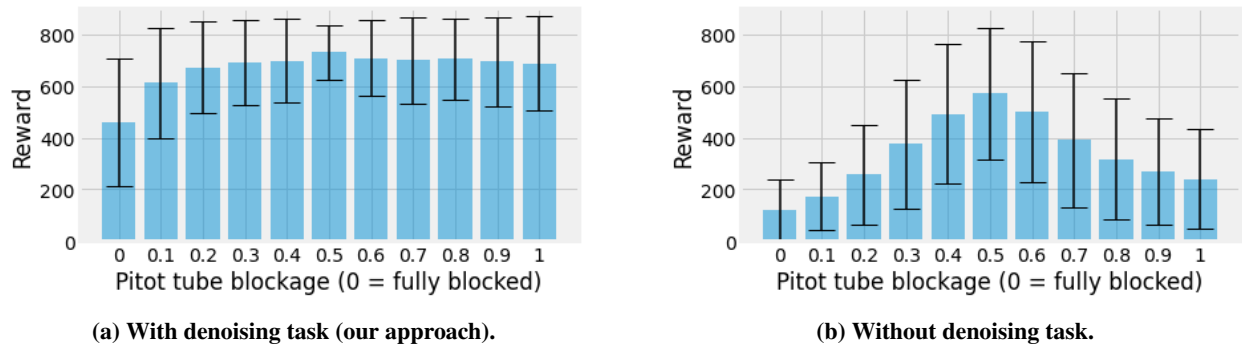


Fig. 8 Comparison of an RL model trained with denoising autoencoder task and without. Showing the mean reward over 500 episodes for different Pitot tube blockage levels.

VII. Conclusion

This paper presented a novel combination of an RL agent trained with an overcomplete denoising autoencoder in the same neural network for improving situational awareness in the cockpit and automatically recover from the stall in the presence of Pitot tube blockages. The combination of Machine Learning methods shows promise in learning meaningful neural connections in the shared network, as the system correctly performed several related tasks. The prediction head of the autoencoder provided reconstructions for all sensor inputs. The results show that the sensor failures caused by a partial Pitot blockage can be reconstructed with low errors. For example, the RMSE for the airspeed reconstruction was below 3.6 knots in the training set and below 6 knots in the testing set (stall scenarios) for any Pitot tube blockage level. This ability of accurate reconstruction can then further be used as a basis for other systems. In this paper an alerting system was built which showed that failures caused by a blocked Pitot tube can be detected on average after 1.4 seconds with 0.98 accuracy, 0.98 recall, 0.97 precision in the training set. For the stall scenarios the detection of failures was instant with an average detection delay of 0.0246 seconds and an accuracy of 0.93. Implementing an autoencoder in this way provides insight into which sensor measurement is incorrect and an estimate of the actual recovery values of the measured readings in real-time. It was shown experimentally that learning this reconstruction task contained very useful information for successfully directing flight controls and guidance systems during situations containing data errors. Therefore, an RL agent was successfully trained using these same neural connections for not only flying whilst experiencing data errors but also to avoid or recover from stalls. The RL agent successfully reduced angle of attack, increased airspeed and returned to the initial altitude. It was shown that the RL system could perform this stall recovery whilst experiencing different variations of a partially blocked Pitot tube.

This research offers a single solution to multiple problems which were previously approached individually. Being

able to accurately perform different tasks with the same neural network can give reassurance that the weights in the neural network are correctly understanding and representing the environment. For example, a decrease in performance of one task can indicate why or if another output of the network could be under-performing. This information is generally rarely accessible in neural networks and is therefore a valid reason why having only one neural network can be of advantage.

It is also pertinent to acknowledge the relation to model-based RL techniques, which also leverage learned models of the environment. Therefore, it would be of value for future work to explore comparative studies between our approach and model-based techniques to address strengths and limitations to further enhance the robustness and applicability of our proposed framework.

It was also shown that learning to do related tasks improves each task individually, which is historically controversial yet an ongoing trend in Machine Learning. A limitation of the proposed system is that when an increasing amount of information is lost, there is no indication on how well the reconstruction and thus the alerting systems works. Therefore, future work is proposed on producing estimates of how reliable the predictions of the neural network are, given information loss from more data errors. This would be in line with the current aim, to use AI to reduce the occurrence of loss of control experienced by pilots.

References

- [1] The Honourable Company of Air Pilots, “Cockpit Automation - Advantages and Safety Challenges,” <https://skybrary.aero/articles/cockpit-automation-advantages-and-safety-challenges>, 2021. Accessed: 2024-01-04.
- [2] Anonymous, “Final Aviation Accident Report Birgenair Flight ALW-301,” Junta Investigadora de Accidentes Aereos, 1996. URL https://www.faa.gov/sites/faa.gov/files/Birgenair301_Accident_Report.pdf.
- [3] Anonymous, “Final Report On the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris,” Bureau d’Enquêtes et d’Analyses pour la sécurité de l’aviation civile, 2012. URL <https://bea.aero/docspa/2009/f-cp090601.en/pdf/f-cp090601.en.pdf>.
- [4] Anonymous, “Final Aircraft Accident Investigation Report PT. Lion Mentari Airlines Boeing 737-8 (MAX),” Komite Nasional Keselamatan Transportasi Republic of Indonesia, 2019. URL http://knkt.dephub.go.id/knkt/ntsc_aviation/baru/2018%20-%20035%20-%20PK-LQP%20Final%20Report.pdf.
- [5] Anonymous, “Interim Investigation Report on Accident to the B737-8 (MAX) Registered ET-AVJ operated by Ethiopian Airlines,” The Federal Democratic Republic of Ethiopia Ministry of Transport Aircraft Accident Investigation Bureau, 2020. URL https://bea.aero/fileadmin/user_upload/ET_302__B737-8MAX_ACCIDENT_FINAL_REPORT.pdf.
- [6] Accardo, D., de Alteriis, G., Conte, C., Rufino, G., Moriello, R. S. L., and Alvarez, O. H., *Advanced Technique to Detect Air Data System Failure by means of MEMS Inertial Sensors*, 2022, p. 0856. <https://doi.org/10.2514/6.2022-0856>.

- [7] Zhao, Y., Zhao, H., Ai, J., and Dong, Y., "Robust data-driven fault detection: An application to aircraft air data sensors," *International Journal of Aerospace Engineering*, Vol. 2022, 2022. <https://doi.org/10.1155/2022/2918458>.
- [8] Ahmad, M. W., Akram, M. U., Ahmad, R., Hameed, K., and Hassan, A., "Intelligent framework for automated failure prediction, detection, and classification of mission critical autonomous flights," *ISA transactions*, Vol. 129, 2022, pp. 355–371. <https://doi.org/https://doi.org/10.1016/j.isatra.2022.01.014>.
- [9] Berri, P. C., Dalla Vedova, M. D., and Mainini, L., "Learning for predictions: Real-time reliability assessment of aerospace systems," *AIAA Journal*, Vol. 60, No. 2, 2022, pp. 566–577. <https://doi.org/https://doi.org/10.2514/1.J060664>.
- [10] Bell, V., Rengasamy, D., Rothwell, B., and Figueredo, G. P., "Anomaly Detection for Unmanned Aerial Vehicle Sensor Data Using a Stacked Recurrent Autoencoder Method with Dynamic Thresholding," *arXiv preprint arXiv:2203.04734*, 2022. <https://doi.org/https://doi.org/10.48550/arXiv.2203.04734>.
- [11] Dhakal, R., Bosma, C., Chaudhary, P., and Kandel, L. N., "UAV Fault and Anomaly Detection Using Autoencoders," *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, 2023, pp. 1–8. <https://doi.org/10.1109/DASC58513.2023.10311126>.
- [12] Campbell, N. H., Grauer, J. A., and Gregory, I. M., *Loss of Control Detection for Commercial Transport Aircraft Using Conditional Variational Autoencoders*, 2021, p. 0778. <https://doi.org/10.2514/6.2021-0778>.
- [13] Li, H., Sang, Y., Ge, H., Yan, J., and Li, S., "Anomaly detection of aviation data bus based on SAE and IMD," *Computers & Security*, Vol. 137, 2024, p. 103619. <https://doi.org/https://doi.org/10.1016/j.cose.2023.103619>.
- [14] Kilic, U., and Unal, G., "Aircraft air data system fault detection and reconstruction scheme design," *Aircraft Engineering and Aerospace Technology*, Vol. 93, No. 6, 2021, pp. 1104–1114. <https://doi.org/10.1108/AEAT-01-2021-0018>.
- [15] Swischuk, R., and Allaire, D., "A machine learning approach to aircraft sensor error detection and correction," *Journal of Computing and Information Science in Engineering*, Vol. 19, No. 4, 2019, p. 041009. <https://doi.org/10.1115/1.4043567>.
- [16] Tomar, D. S., Gauci, J., Dingli, A., Muscat, A., and Zammit-Mangion, D., "Automated Aircraft Stall Recovery using Reinforcement Learning and Supervised Learning Techniques," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, IEEE, 2021, pp. 1–7. <https://doi.org/10.1109/DASC52595.2021.9594316>.
- [17] Cao, H., Zeng, W., Jiang, H., Hu, H., Li, C., Lu, W., and Xiong, H., "Two-Stage Strategy to Achieve a Reinforcement Learning-Based Upset Recovery Policy for Aircraft," *2021 China Automation Congress (CAC)*, IEEE, 2021, pp. 2080–2085. <https://doi.org/10.1109/CAC53003.2021.9727381>.
- [18] Lang, X., Cen, F., Li, Q., and Lu, B., "Deep reinforcement learning-based upset recovery control for generic transport aircraft," *Aerospace Systems*, 2022, pp. 1–10. <https://doi.org/10.1007/s42401-022-00156-8>.
- [19] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. <https://doi.org/https://doi.org/10.48550/arXiv.1707.06347>.

- [20] Berndt, J., “JSBSim: An open source flight dynamics model in C++,” *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004, p. 4923. <https://doi.org/https://doi.org/10.2514/6.2004-4923>.
- [21] Berndt, J., and the JSBSim Development Team, “JSBsim An open source, platform-independent, flight dynamics model in C++,” <https://jsbsim.sourceforge.net/documentation.html>, 2011. Accessed: 2021-12-20.
- [22] Anonymous, “Air Data Handbook,” Goodrich Cooperation, Rosemount Aerospace, Collins Aerospace, 2002.



DSPACE

<https://dspace.org/>

AI for real-time tolerance to critical flight data errors in large aircraft

Koopman, Cynthia; Zammit-Mangion, David

2023-06-08

American Institute of Aeronautics and Astronautics (AIAA)

<https://doi.org/10.2514/6.2023-4327>

Koopman C, Zammit-Mangion D. (2023) AI for real-time tolerance to critical flight data errors in large aircraft. In: AIAA AVIATION 2023 Forum, 12-16 June 2023, San Diego, USA, pp. 4327. Paper number AIAA 2023-4327