Mechanical Systems and Signal Processing, Volume 171, May 2022, Article number 108873
DOI:10.1016/j.ymssp.2022.108873

Application of Deep Reinforcement Learning for Extremely Rare Failure Prediction in Aircraft Maintenance.

Maren David Dangut, **Ian K. Jennions, ***Steve King, * Zakwan Skaf**

*,**,***Integrated Vehicle Health Management Centre (IVHM),

Cranfield University Bedfordshire MK43 0AL, United Kingdom

****Mechanical Engineering Department, Higher Colleges of Technology, United Arab Emirates

*maren.dangut@cranfield.ac.uk, **i.jennions @cranfield.ac.uk, ***s.p.king@cranfield.ac.uk, ****zskaf@hct.ac.ae

Abstract

The use of aircraft operational logs to predict potential failure that may lead to disruption poses many challenges and has yet to be fully explored. Given that aircraft are high-integrity assets, failures are extremely rare, and hence the distribution of relevant log data containing prior indicators will be highly skewed to the normal (healthy) case. This will present a significant challenge in using data-driven techniques because the model will be biased to the heavily weighted no-fault outcomes. This paper presents a novel approach for predicting unscheduled aircraft maintenance action based on deep reinforcement learning techniques using aircraft central maintenance system logs. The algorithm transforms the rare failure prediction problem into a sequential decision-making process that is optimised using a reward system that penalises proposed predictions that result in a false diagnosis and preferentially favours predictions that result in the right diagnosis. The validation data is directly associated with the physical health aspects of the aircraft components. The influence of extremely rare failure prediction on the proposed method is analysed. The effectiveness of the new approach is verified by comparison with previous studies, cost-sensitive and oversampling methods. Performance was evaluated based on G-mean and false-positives rates. The proposed approach shows the superior performance of 20.3% improvement in G-mean and 97% reduction in false-positive rate.

Keywords: Extremely rare event, deep reinforcement learning, imbalance classification, aircraft maintenance.

Nomenclature

Aircraft Central Maintenance System	ACMS
Air traffic service	1TX1
Autoencoder Bidirectional Gated Recurrent Unit	AE-BGRU
Avionics equipment ventilation computer	10HQ
Convolutional Neural Network	CNN
Deep Reinforcement Learning	DRL
Deep Q-Network	DQN
Double Deep Q-Network	DDQN
Double Deep State Action Reward State Action	DDSARSA
Extreme Value Analysis	EVE
Electronic Control Unit/ Electronic Engine Unit	400KS
False Positive Rate	FPR
False Negative Rete	FNR
Functional Identification Number	FIN
Feature engineering	FE
Flow control valve	11HB
Gated Recurrent Unit	GRU
Geometric Mean	G-Mean
High-Pressure Bleed Valve	4000HA
Long Shot-Term Memory	LSTM
Markov Decision Process	MDP
No-fault found	NFF
Peak Over Threshold	POT
Performance Report	PR
Prioritized Experience Replay Memory	PER
Random Forest	RF
Reinforcement Learning	RL
Satellite Data unit	5RV1
State Action Reward State Action	SARSA
Synthetic Minority Oversampling Technique	SMOTE
Quick Access Recorder	QAR

1. Introduction

In recent times, the concept of predictive maintenance has continued to advance, especially in a complex system such as an aircraft. Predictive maintenance is designed to monitor the health condition of in-service equipment and to forecast maintenance needs. It provides a cost-benefit compared to time-based approaches such as preventive maintenance because maintenance is carried out only when needed [1]. As the popularity of predictive maintenance models increases in the aviation industry, one of the critical challenges is dealing with unplanned failures, i.e. rarely reported events. In other words, the challenge of learning from an extremely imbalanced dataset using standard machine learning algorithms.

Furthermore, using the data from operational equipment logs to develop predictive models poses many challenges that have not yet been fully explored, as logs are mainly used for anomaly detection and debugging failure. The logs generated in complex systems such as aircraft are mostly multivariate time series (multiple interrelated streams of data are recorded simultaneously). This type of data is commonly recorded from several monitoring systems, such as the condition-based or sensors, collected over time. They may, therefore, be regarded as complex multivariate time-series data. Given that aircraft are high-integrity assets, failures are extremely rare, and hence the distribution of relevant data containing prior indicators will be highly skewed to the normal (healthy) case. This will present a significant challenge in using data-driven techniques to 'learning' relationships/patterns that depict fault scenarios since the model will be biased to the heavily weighted no-fault outcomes.

Some of the characteristics of a system log that cause a challenge in predictive modelling are:

(i) Heterogeneous in nature containing symbolic sequences, numeric time-series, categorical variables, and unstructured text. It is a non-trivial task to translate free-text log messages into meaningful features.

(ii) System log volume can be large in complex systems, which poses computational challenges.

(iii) Having a rare occurrence of failure results in a lack of enough information to anticipate certain specific families of faults.

Thus, this study investigates the use of aircraft operational log-based data to develop a predictive model for rare failure prediction in aircraft. Also, to determine which variables are likely to indicate the target failures. An issue of predictive maintenance lies in the rigid nature of data (data changing over time). If correct parameters are not built-in, it can risk incorrect forecasts and erroneous 'fault' messages. For instance, based on historical behaviour, if a maintenance operator forecasts that a component will fail within 100 flights, they might schedule removal to prevent operation failure.

However, upon removal, the part may test as no fault found (NFF), costing the operator unnecessary time and money. Therefore, developing a robust predictive model is necessary, especially for safety-critical equipment such as aircraft.

In order to make use of log-based data to develop a robust predictive maintenance model, generally, the first step is to interpret the logs, filter out a large amount of noise (that is, data irrelevant to the set goal), and extract predictive features. Also, the known failure cases need to be collected for learning and evaluation. The problem needs to be transformed into an appropriate learning scenario, and a performance measure that reflects real-world needs must be determined. Figure 1 shows the proposed process of discovering knowledge from raw data. The raw heterogeneous and multivariate data collected from different sources is stored in a database. The raw data usually contains many analytical challenges requiring pre-processing, such as data incompleteness, lack of example behaviours and trends, missing or null values, lack of exact features of interest, and noise. Data pre-processing and transformation (into a suitable format for machine learning) occurs in Stage 2 of Figure 1. A feature engineering (FE) process is carried out at stage three; it helps collect relevant features related to the desired goal. FE is the integral and critical step of the machine learning process because the quality of data and the right features contribute majorly to a predictive model's performance. After the pre-processing and FE phase, the data is divided into training and validation. Stage four is where the machine learning algorithm for pattern recognition or classification is trained using the training data. The model is then evaluated at stage five. The outcome can then give insightful knowledge for more informed decision-making.

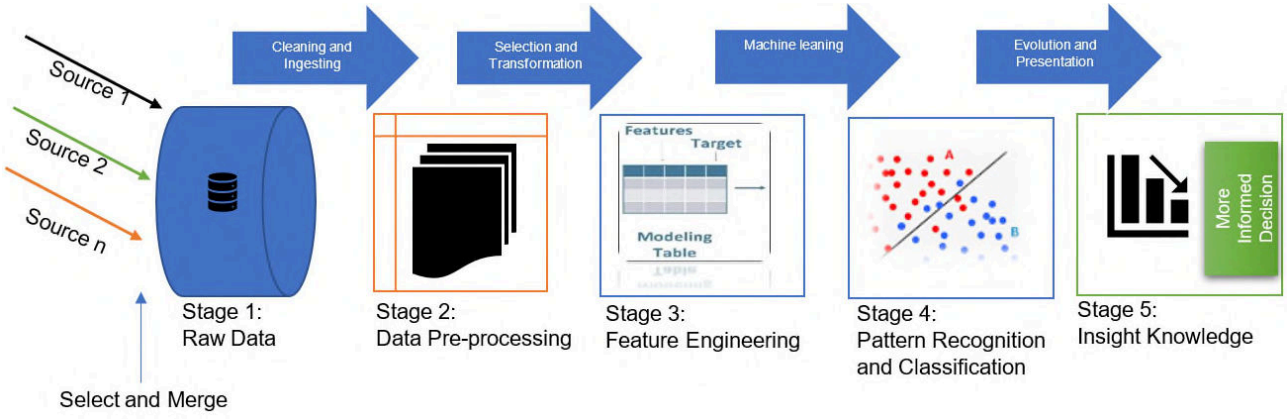


Figure 1. Basic Data Knowledge Discovery Process

In a rare failure prediction problem, the pre-processed dataset usually has a skewed distribution. For example, in the ACMS dataset, the non-failure represent negatively labelled samples, and the failure represent positively labelled samples. The negative samples far outnumber the positively labelled, causing the data to be highly imbalanced. The disproportion between classes can be very low (e.g.

5% or less). Various solutions for the slight rare failure problem (say proportions of 40:60 to 30:70) have been suggested in the literature. However, in a situation where the imbalance ratio is extreme, say less than or equal to 5%, the problem becomes more challenging to handle [2][3]. In such a scenario, the standard approaches for normal failure prediction (such as statistical approaches, traditional machine learning algorithms and associated rules) become limited [4–6]. The reason is that most normal failure patterns are similar to each other and are substantially represented.

In contrast, rare failure is typically one-of-a-kind, and hence it becomes difficult to learn temporal patterns using traditional machine learning approaches. That is why many aircraft predictive maintenance models are based on simple "threshold" monitoring rules capable of detecting only simple faults and, consequently, having high false-positive rates (FPR)[7]. Hence, it is vital to provide an accurate prediction of failures and, at the same time, have a very low FPR. That can improve the effectiveness of the aircraft health monitoring systems and, in turn, enhance the availability of the aircraft.

This study considers the case of developing a model to predict unplanned failure and replacement of aircraft components. The dataset used contains extremely rare failures of the target component. The imbalance ratio for each target component is less than 3% of the total dataset, making it difficult to develop a predictive model effectively using the existing traditional machine learning approaches. Therefore, this study aims to show the applicability of deep reinforcement learning for training an extremely rare failure predictive model instead of the widely used machine learning or deep learning methods for slightly rare failure predictions. The proposed model is trained using a real-world aircraft central maintenance system (ACMS) dataset.

The proposed approach considers the problem of extremely rare event prediction from a reinforcement learning point of view. The problem is formulated as a Markov sequential decision-making process and solved by combining reinforcement learning with deep neural networks. The approach enables the model to remember a long sequence of failure patterns. The reward function is specifically constructed to counter agent bias towards the majority class during model training. Figure 2 shows the interaction between the elements of reinforcement learning. Here, the agent-classifier takes action in an environment; transition through the time series ACMS dataset is considered an environment in the proposed approach. A reward is returned based on the action taken (classify pattern as fault or non-fault) at a given state.

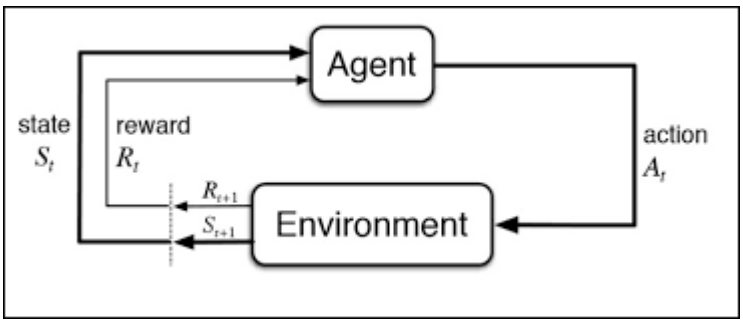


Figure 2 Visual representation of iterative feedback loop of actions, states, and rewards in reinforcement learning

Rationale: DRL algorithms were traditionally designed for performance optimisation with very large input space [8]. Therefore, exploring the application of DRL approaches for complex systems large log-based datasets can significantly benefit the predictive maintenance, especially that data is continually increasing in dimension [9]. The rationale for the proposed method is to explore the applicability of deep reinforcement learning for extremely rare prediction problems, purposely for performance optimisation in complex systems predictive maintenance models, to minimize downtime and increase the utilization rate of the vehicles or components. The motivation for the possible performance improvement in the proposed algorithm is the combination of the convolutions in deep neural networks that enhance learning relationships between variables in the dataset. Also, the reward function, which helps to counter bias during model training and prioritised experience replay memory, which instead of uniformly sampling transactions from replay memory, employs a prioritised approach that also entails replaying the important transactions more frequently. Hence, optimising the learning process. Also, DRL uses a reward function to optimise future rewards, in contrast to a machine learning (regression or classification) model that predicts the probability of future outcomes. Therefore, it can be assumed that deep reinforcement learning methods are ideally best for imbalanced classification problems because of its learning mechanism and specific learning environment and reward function.

This paper presents a novel approach using deep reinforcement learning techniques to predict unplanned aircraft maintenance actions using data from operational flight logs and maintenance report information. The approach first identifies relevant temporal patterns that correspond to each component failure. It then transforms the problem into a sequential decision-making process that is optimized using deep reinforcement learning algorithms utilizing a reward system that penalizes proposed predictions leading to a false diagnosis and preferentially favours predictions that lead to a correct diagnosis. The failure messages in the ACMS data is directly associated with physical health aspects of the vehicle, asset or component (such as pressure, vibration, temperature, acoustics, viscosity, flow rate data). The patterns that are input to the algorithm represent the history state of the components, and they are labelled as failure or non-failures. The reward function is specifically

constructed to counter agent bias towards the majority class during model training. The strategy allows adequate handling of extremely imbalanced problems in predictive maintenance modelling. The influence of extremely rare failure prediction on the proposed deep reinforcement learning models is analyzed.

The main contributions of this paper are as follows:

1. To show a novel application of deep reinforcement learning to predict extremely rare failure problems in complex aircraft systems. The new deep reinforcement learning approach is designed to capture the patterns of extremely rare component failures adequately. The model is trained to predict aircraft component replacement well in advance of failure. The technique includes designing and developing an environment for the state-action, a reward function for rewarding agent-classifier actions, and the unique arrangement of a deep neural network architecture for policy optimization.
2. The new method is validated using a real-world aircraft central maintenance system dataset. Exploring the ACMS dataset for developing a predictive maintenance model is a significant contribution because of its heterogeneous nature, challenging to analyse.

The rest of this paper is organized as follows. Section two provides related work. Section three presents the proposed new method and its implementation. Section four presents the case study. Section five shows the results and discussion, and the conclusion is presented in section six.

2. Related Work

One of the design goals of predictive maintenance is to avoid unexpected failures by monitoring the vehicle condition and providing failure alerts well in advance. Predictive maintenance models are developed to forecast when likely the vehicle will fail, so that maintenance can be systematically scheduled to occur way in advance before the failure point. Predictive maintenance can be modelled in three ways: physics-based, knowledge-based, and data-driven-based [10]. Physics-based modelling can be defined as a simplified mathematical description of a system or process to assist calculations and predictions [11]. The prediction is based on a mathematical equation inside the mode; therefore, it uses a limited amount of data compared to other methods. However, the physics-based model is challenging to create and implement, especially for complex systems, because it is sensitive to the system's design and material properties. Also, enough component information and a good knowledge of the failure mechanism is highly required to formulate the model.

The knowledge-based model, also known as the expert system, uses defined rules or fuzzy logic to solve complex problems. The rules are set based on the knowledge of a domain expert. Converting

domain knowledge to a set of rules is challenging, which requires another technique for prognostics. Also, the set of rules needs to be updated anytime there is any system update. This process can be cumbersome and sometimes impractical, especially in a complex system with many components and processes.

The data-driven approach involves training machine learning algorithms using large historical datasets to learn a system behaviour model automatically. A data-driven approach is easy to implement, flexible, adaptable with a low cost of implementation. However, large historical data representing failure is needed, and getting such data is always challenging. However, the advancement in technology data is increasingly available, making it more appealing to use a data-driven approach for developing predictive maintenance models in complex systems. For performance optimization a hybrid of the two or three approaches can be investigated [12], which is one of the study's main objectives.

2.1 Rare Event prediction

The challenge of predicting rare events has been around for some time and is still an ongoing research area [1]. Many solutions have been proposed in the literature, especially related to the maintenance of heavy industrial equipment and other domains that require rare event prediction. The existing solutions are primarily found in two groups: statistical methods and machine learning methods. Examples of statistical methods are the extreme value theory or extreme value analysis (EVA) [13,14] and the peak over threshold (POT) methods [15]. These methods deal with extreme deviation from the mean of a probability distribution in a dataset [16,17]. Statistical methods draw population inferences from a sample, whereas machine learning finds generalizable predictive patterns [18]. Machine learning approaches are desirable in this study because they are particularly helpful when harnessing knowledge from large heterogeneous datasets. They are more effective and efficient compared to other data mining and analysis methods.

Machine learning approaches are divided basically into supervised, unsupervised, and reinforcement learning. Other hybrid learnings are semi-supervised, self-supervised and multi-instance learning. Supervised learning techniques involve learning or inferring using labelled training datasets. An example of supervised learning is seen in building a model for rare event prediction based on labelled data (the training set) [19]. One of the strongest advantages of supervised methods is that they can easily be validated, but the training data must be labelled.

On the other hand, unsupervised learning involves developing models using unlabelled datasets; this is mainly used for problems such as anomaly detection, deviation detection, outlier analysis, and exception mining. These methods analyse each event one after another to determine how similar or

dissimilar they are to the majority. Their success depends on the choice of parameters, such as similarity measures and dimension weighting. Therefore, because the dataset used in this study has defined labels, the supervised machine learning approach is considered.

Furthermore, rare event prediction can also be modelled using association rules (knowledge-based). However, this approach is more effective for a small and simple system [20], not the large heterogeneous datasets studied here. The use of associative rules for a large and complex system is quite challenging and, in some cases, impractical because domain experts need to continually update the rules in the event of any upgrades or changes, which is time-consuming and cumbersome [21][22]. Another potential approach is reinforcement learning which can be considered from a sequential learning point of view. In this type of learning, an agent takes the best actions sequentially in a particular environment in order to maximise cumulative rewards [23]. The current study focuses on the deep reinforcement learning approach.

Why is deep reinforcement learning considered for extremely rare event prediction instead of the standard deep learning or machine learning approach? It is a legitimate question, and the answer is subjective. Existing machine learning algorithms can handle the data imbalance problem in diverse dimensions depending on the type of dataset. However, considering that a situation where the target events are extremely rare, those methods become limited [3,4,24]. For instance, an imbalanced classification problem can be handled at the data level either by under-sampling the majority (negatively labelled) samples to balance with the minority class (positively labelled) or over-sample the minority class by creating more synthetic samples. Then the model can be trained using any existing machine learning algorithm. In this case of under-sampling, if the imbalance ratio is say1:200, in a total of a million records, about 0.5% will remain in the positively labelled dataset. After under-sampling, a total of approximately 1% of the original dataset will be left. The standard machine learning algorithms (such as Support Vector Machine, Decision Tree or Random Forest) can be used to train the model with data of this size. However, the potential information in the remaining ~99% of data left out will not be utilized, producing a low-sensitivity model [25].

Another approach could be to over-sample the minority class, then use machine learning to train the model. This approach has the drawback of increasing the likelihood of overfitting since it replicates the minority class examples. The Synthetic Minority Oversampling Technique (SMOTE) [26] has been developed to mitigate overfitting in random oversampling by taking a subset of data from the minority class as an example and then creating new synthetic similar instances. However, SMOTE has the drawback of not considering neighbouring examples from other classes when generating synthetic samples. That can cause overlapping of classes and can also introduce additional noise into the training data. SMOTE is also ineffective in high dimensional data, as argued by Lusa et al. [27].

In recent times, many solutions have been proposed to correct the drawbacks of SMOTE[28–30] and other novel solutions which are specific to either the application domain or dataset in question, as presented by Alberto et al. [25].

Furthermore, another approach is to transform the dataset and then uses deep learning methods to train the model. Recent examples of time-series-based deep learning models have been proved to provide state-of-the-art performance in handling slightly rare event prediction problems. For example, the combination of an Auto-encoder with LSTM or GRU deep neural networks has been shown in Maren et al. [31] and Di et al.[32]. Although these models have continued to improve over time, the challenge of handling an extremely imbalanced dataset, or extremely rare event prediction, remains an area that requires continuous improvement. For instance, model performance degradation is seen in training deep neural networks with an imbalanced dataset. Deep learning methods are affected by a highly imbalanced dataset because the overall total error cost representing the majority samples impacts the minority class samples by overwhelming the gradient responsible for updating the model's weights. Hence creating a biased model that will produce a high FPR [31,33]. Therefore, the open literature lacks a unified solution to handling extreme imbalance classification problems, especially for large heterogeneous ACMS datasets. Hence, this study seeks to provide a solution to an extreme imbalance problem using a deep reinforcement learning (DRL) approach. The solution aims to optimise the data-driven model's performance by avoiding biases and reducing the false positive rate.

2.2 Deep reinforcement learning for predictive model

The integration of deep learning with reinforcement learning, known as DRL, to optimise model performance is gaining more research attention, and it is producing state-of-the-art solutions [34]. For instance, the integration of deep learning and reinforcement learning has led to the emergence of a novel technique called the deep Q-network (DQN)[3,23,35]. DRL has made the application of reinforcement learning attractive in different domains. One such domain is in developing predictive maintenance models for complex systems. A detailed survey on deep reinforcement learning and its applications can be found in a study by Kia et al. [23]. The DRL application can be seen in robotics and gaming [30][31], where different techniques are used to achieve the desired results. Also, in communication and networking [36], detecting and predicting failure notes in the network and cyber security [37] for detecting fraudulent events in the system. In the financial sector, DRL is used for solving complex business problems [38][39] and for inventory management and resource allocation [40]. Others are in medicine [41], engineering and manufacturing [42][43].

Recently, the application of DRL for equipment maintenance is gaining more research attention. A study by Knowles et al. [44] has shown how to integrate reinforcement learning into condition-based maintenance. Rocchetta et al. [45] developed a framework based on DQN to optimise power grid equipment's operation and maintenance. Both approaches are based on Markov Decision Process (MDP) and DRL. The applicability of deep reinforcement learning for equipment health indicator learning is also shown in a study by Chi Zhang et al. [46]. However, the open literature lacks any exhaustive study that shows how extremely rare event prediction in complex systems can be modelled using deep reinforcement learning approaches, which our study seeks to fill.

The current study is motivated by the fact that exploring the application of DRL methods for real-world problems, such as rare equipment failure prediction, for potential performance optimization opportunities. In data classification problems, DRL has served better in removing noise from data and learning hard temporal features, improving predictive models' performance [16]. Lin et al. [3] pointed out that deep reinforcement learning methods are ideal for imbalanced classification problems because of their learning mechanism and specific training environment and the control of the learning process using reward function. DRL uses a reward function to optimize future rewards, in contrast to a machine learning (regression or classification) model that predicts future outcomes probability.

The DRL framework can be constructed by combining a deep neural network and reinforcement learning. That can be seen in Q(λ)-learning [47], where the reward function can give a high reward or a penalty for an action taken by the agent-classifier on a positively labelled class (minority). With more attention given to the minority class, the algorithm can respond favourably to both classes during learning, hence enhancing the resulting model's effectiveness.

As demonstrated by this review of the open literature, research on the application of deep reinforcement learning for extreme rare event prediction in complex systems is limited. Thus, this paper demonstrates the application of deep reinforcement learning in aircraft predictive maintenance modelling, focusing on developing a model to predict extremely rare failure using a heterogeneous log-based ACMS dataset.

3. Methodology

3.1 Description of reinforcement learning based on the Markov Decision Process

In reinforcement learning and Markov Decision Process (MDP), the agent interacts with an environment $\mathcal{E}$ sequentially over a discrete-time step t. The agent takes action a_t at time t after observing the state s_t. Based on the agent's action a_t, reward r_t is returned. The process can be

represented as a 7-tuple of $\boldsymbol{M} = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{s_0}, \boldsymbol{\gamma}, \boldsymbol{T})$, where S is the set of states. A is the set of actions. P is the transition probability distribution represented as $(P: SAS \rightarrow R^+)$. R is the reward function, represented as $R: SA \rightarrow R$ and R^+ a returned immediate reward received after transitioning from state s to next state s', due to action a. s_0 is the initial state distribution defined as $s_0: S \rightarrow R^+$. γ is the discount factor $\gamma \in [0,1]$, a lower discount factor motivates the decision-maker to favour taking actions early rather than postponing them indefinitely. **T** is the transitional probability distribution.

Once the MDP is defined, the target is to have an agent that can determine, at state s_t, which best next action to take in order to maximize the reward r_t. A gradient descent function can be used to maximize the reward based on a defined policy π_θ. For example, the agent takes an action $\hat{y}_t \in A$ with respect to the optimal policy $\pi(\hat{y}_t|s_t)$: $SA \rightarrow R^+$ and observed reward r_t for that action. The cumulative discount sum of the rewards is the objective function optimized by the policy π_θ. The optimal policy is created using a value function, which is a defined estimated value related to each state. The value function can either be a V-function [48], which estimates the value for each state, or the Q-function [48], which estimates the value for each pair of state-action $Q(s, a)$. The basic transaction of Q-learning keep a lookup table, in contrast to the deep Q-networks which leverages the use of replay memory to store trajectory transactions and the stored interaction are fetched from the replay memory in mini-batches to train the deep neural networks [8]. In other words, deep Q-learning fits the Q-function with deep neural networks.

MDP based models are used for planning future action and rewards. Methods of solving reinforcement problems based on planning are either model-based or model-free. The model-based technique is when transitional probability T and reward R are known. In this case, the optimization process can learn from T and R. The model-free approach is when T and R are unknown. In that case, the optimization process will directly learn the best policy without knowing T and R using trial-and-errors learners [49]. In our implementation, we adapt a State Action Reward State Action (SARSA) learning and Deep Q-network (DQN) methods [50][47] which are based on a model-based reinforcement learning approach. SARSA is an on-policy model meaning the agent gets the optimal policy and uses it to act, while Q-learning is off-policy because it estimates the reward for future action and appends a value to the new state without using any greedy policy [50].

3.2 Formulation of Rare Failure Prediction Framework Based on Markov Decision Process

To formulate the DRL-based rare failure prediction approach using the log-based ACMS dataset. The problem is considered as a sequence-to-sequence learning process, where the agent serves as a classifier. The agent receives patterns proceeding with each failure sequentially and classifies each

pattern as either failure or non-failure. The environment then returns a reward based on the agent's action. A positive reward is returned if the agent makes a correct classification; otherwise, a negative reward is returned. In the process, the agent will learn optimal behaviour from the environment and subsequently improve the agent classification accuracy.

Assume the training dataset is

$$D = \left\{\left[\left(x_{1,1}, x_{1,2} \dots x_{1,n}\right), \left(y_1\right)\right], \left[\left(\left(x_{2,1}, x_{2,2} \dots x_{2,n}\right), \left(y_2\right)\right)\right], \dots, \left[\left(\left(x_{\mathrm{m},1}, x_{\mathrm{m},2} \dots x_{\mathrm{m},n}\right), \left(y_{\mathrm{n}}\right)\right)\right]\right\},$$

Where $x_{i,j}$ is the failure pattern and y_i[s] the labels.

Table 1 shows the sample of the data and the interaction. The training dataset contains n-number of features and their corresponding labels. To transform the data for the DRL application pattern related to each target event with its corresponding labels is considered as state S. At every given state, the agent-classifier takes action by considering patterns related to each event as inputs and then performing a classification action a at time t. Based on the action taken, a reward r_t is returned. At the end of each trajectory, a cumulative reward R_t is returned, and the transaction is recorded in a replay buffer.

Table 1. Representation of interaction of the agent with the environment

n-Features					**Labels**	**Agent classifier**
	x_1	x_2	x_3	x_n	y_i	-
The pattern of event 1	S_t				a_t	$\leftarrow r_t$
The pattern of event 2	S_{t+1}				a_{t+1}	$\leftarrow r_{t+1}$
…	…				…	…
The pattern of event n	S_{t+n}				a_{t+n}	$\leftarrow R_n$

A window is defined using the flight leg, and the end of each window is considered a trajectory. The agent-classifier can learn which action is favourable at a future given state by taking action and receiving a returned reward. During the training, because of the rarity of target events, the trained Q-network will favour the majority class more than the minority (also referred to as the data imbalance problem). A reward function is defined to control the biases during learning by assigning different rewards for various classes present. That will handle the challenge of the extreme imbalance in the dataset.

In order to train the model on the ACMS dataset, the following DRL model parameters are defined as follows.

Observation Space (S): contains all variables the agent-classifier needs to consider before classifying a data point as either positive or negative. For the problem under consideration, the agent is expected to see all the pattern variables before making a decision. The intuition here is at each given time-step, the agent-classifier is expected to consider the previous, present and future patterns before updating its weight. At the start of the training, the agent-classifier receives the first pattern as a sequence of failure/warning messages. The order of sequence is maintained so as not to alter the pattern leading to equipment maintenance. The input is in the form of a 3D array (Samples, Time Steps, and Features)

Action Space (A): The agent classifier takes action once it has assessed the environment. In our case, the action is binary $A = \{1, -1\}$ classified as positive or negative corresponding to the labels in the training dataset.

Reward (R): Represented as r_t , the reward is returned based on the action taken by the agent classifier on the environment. If the agent predicts the given pattern correctly as positive, a high reward will be returned. If it misclassifies, a penalty is given in the form of a negative value. To improve the prediction of the minority class, at each time step, a reward function is defined so that a higher reward is returned for the correct classification of the minority class and larger penalties for misclassification. This helps the agent-classifiers to become less biased towards the majority class. The reward values are chosen using the imbalance ratio defined in equation 1.

$$r_t = \begin{cases} \lambda\rho\,, & a_t = y_t \ \ where\ s_t \in D_N \\ -\lambda\rho\,, & a_t \neq y_t \ \ \ where\ s_t \in D_N \\ 1\,, & a_t = y_t\ where\ s_t \in D_p \\ -1, & a_t \neq y_t \ \ where\ s_t \in D_P \end{cases} \tag{1}$$

where $\rho = \frac{D_P}{D_N}$, D_N is the given number of majority class elements and D_P the given number of minority class elements. λ is a trade-off parameter that allows the control of the composite to be between speed and accuracy. Where $\lambda \in [0,1]$. The range of the grid search for the parameter lambda (λ) is define in the range [0,1]. The dynamic adjustment of the reward function hyperparameters is achieved by the use of a defined function. The function is designed as part of the reward, it allows a user to specify the upper and lower values for the lambda (λ) for the model to test. The model iterate and measures the best value of lambda (λ).

Transition probability distribution dynamics (T): is the probability of transitioning from one state S_t to another state s_{t+1} in a single step, $p(s_{t+1}|s_t, a_t)$. In our case, it is deterministic the agent classifier moves from a current state s_t to the next state s_{t+1} in the sequence of patterns in the dataset.

Discount factor (γ) : The factor $\gamma \in [0,1]$, is the weight of importance of future rewards. The discount factor needs to be defined carefully since we are considering a sequence to sequence approach where a successive pattern can be related.

Exploration rate: The rate $\varepsilon = [0,1]$. It is important to explore as much of the state-action space as possible to achieve optimal policy. Therefore, we choose the e-greedy approach [51].

Episode (e): is the transaction trajectory of all the states that came from the initial state to the terminal state. In this solution, an episode defines as when an agent classifier reaches the end of the window.

Policy (π_θ): is a function that receives a sample as input and then returns the probabilities of the label, represented as the mapping function $\pi: s \rightarrow A$ where $\pi_\theta(s_t)$ denotes the action a_t performed by an agent at state s_t. In an MDP, the sequence of (s, a, r) in an episode forms a policy trajectory. End of every episode, a total cumulative reward is returned from the environment.

$$G_t = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \quad (2)$$

The goal of every RL algorithm is to find an optimal policy π^* Which attains the maximum expected return from all states. A policy is an agent-classifier behaviour action, and it specifies what action to take at each step. The stochastic policy is expressed as

$$\pi(a|s) = P(A_t = q, S_t = s) \quad (3)$$

Where $\pi(a \mid s)$ is the probability of taking action a in a state s under a policy π

Experience replay memory: replay memory is used in moderating the effect of the imbalance problem. The replay memory is split equally into sub-memories between classes. After the split, then each corresponding class will be appended in its memory instead of overwriting the minority sample with the overwhelming majority. This approach will ensure that when samples are randomly fetched from memory to train the agent, it will balance all the training dataset classes.

3.3. Implementation of reinforcement learning for extremely rare failure prediction.

As seen in Figure 3, a defined reward function (equation1) is used to provide a known reward for each action at every step. The dataset represents the environment where the agent-classifier takes an action a at a given state s (see Table 1), and based on the action taken, a reward is returned. The DQN addresses the fundamental instability problem of using a functional approximation in

reinforcement learning (RL) by using two techniques: experience replay memory and target networks(θ). Experience replay memory stores transitions of the form Q(s_t, a_t, s_{t+1}, r_{t+1}) in a replay buffer. This enables the agent-classifier to sample from and train on previously observed data. Not only does this massively reduce the number of interactions needed with the environment, but batches of experience can be sampled, reducing the variance of learning updates. Furthermore, the temporal correlations that can adversely affect RL algorithms are avoided by sampling uniformly from a large memory. Finally, from a practical perspective, batches of data can be efficiently processed in parallel by modern hardware, increasing throughput.

The original DQN algorithm used uniform sampling [52]. However, a later study shows that prioritizing samples based on eligibility trace [53] is more effective for learning. Q-learning seeks to find the best action to take for any finite MDP, given the current state. The Q-learning algorithm learns a policy that maximises a cumulative reward under a specific state-action pair Q(s, a). Therefore, within a given trajectory, the algorithm will perform a series of actions to obtain a maximum total reward.

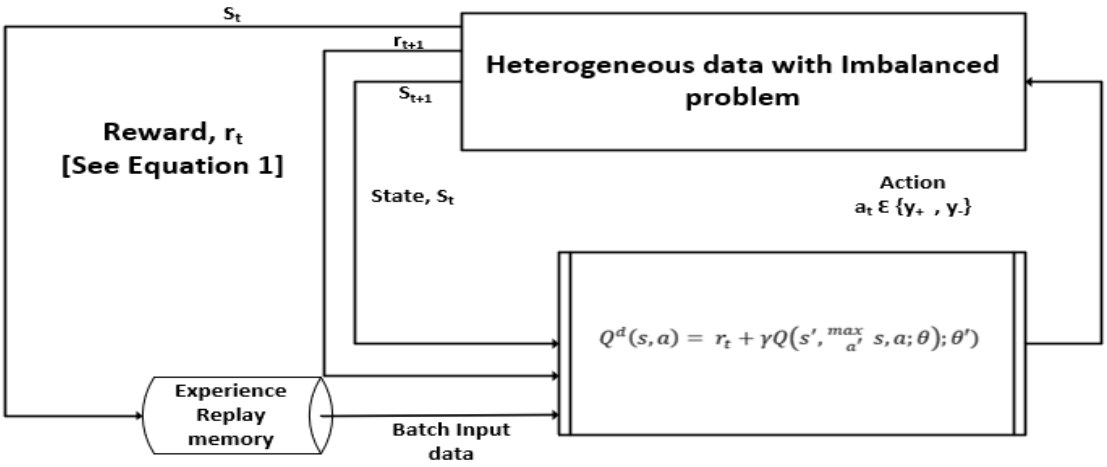


Figure 3. Deep Reinforcement Learning for rare event prediction

I. Deep Reinforcement Learning optimal policy: An optimal policy is an integral part of the proposed DRL algorithm. Basically, in reinforcement learning, a policy is responsible for choosing an action from a given state. Therefore, an optimal policy chooses the best action from a state. Choosing the best policy is the goal of every reinforcement learning algorithm. In the proposed approach, unlike normal reinforcement learning, the agent-classifier receives an environment state as input represented by a training sample and then performs an action (classification) under the control of a policy. DRL-based classification policymaking aims to learn the classification policy that maximizes the total reward during the entire training period.

Finding an optimal policy in Q-learning, a value function is needed, and to calculate the value function a total cumulative reward (G_t) is required. To find G_t a sum of rewards for every action is needed that is

$$r_{t+1} + r_{t+2} + r_{t+3} \ldots = \sum R_t = G_t = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \quad (4)$$

Where T is the Trajectory.

A value function is a function that follows a policy for each step to estimate the expected future reward, expressed as

$$V_{(s)} = \mathbb{E}[G_t | S_t = s] \quad (5)$$

There are two types of value functions; the state-value function (see equation 6) determines an agent's goodness in a given state. The action-value function (equation 7) determines how good it is to perform a given action in a given state.

$V_{\pi(s)} = \mathbb{E}_\pi[G_t | S_t = s]$, the state-value function (6)

$q_{(s,a)} = \mathbb{E}_\pi[G_t | S_t = s, A_{t=a}]$, the action-value function (7)

An optimal policy π^* is the maximum expected reward for each state express as

$$\pi^*(a|s) = \max_\pi Q_\pi(s, a) \quad (8)$$

The next step is to find a method to predict possible future rewards, but the challenge is that possible actions at future time-steps are unknown. Since the Bellman equation helps in calculating Q* at each time step, it gives a way to determine the optimal policy. Therefore, the Bellman equation[54] is used to drive the optimal policy, which incorporates the possible actions' probability at future time-steps.

$$V(s) = \mathbb{E}[R' + \gamma V(S') | S_t = S] \quad (9)$$

Where S_{t+1} *or* S' *is the Next state* and S_t *or* S *is the Current state*. Equation 9 is the Value Function of current state = Immediate reward + value function of the next state. The Bellman for the action-value function becomes.

$$Q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi S' \quad (10)$$

More detail on Bellman's expression for state-value and action-value function is explained by David Silver[55].

Substituting equation (8) in (9) to get

$$Q_\pi(s,a) \quad = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s) Q_{\pi(}(s',a')) \tag{11}$$

As we can infer from equation 11, the optimal policy π* is to take the best action at each given state defined by $Q(s,a)$. Therefore, the optimal Q-function becomes

$$Q^{\pi^*}(s,a) = \max_a \left\{ R_s^a + \gamma \sum_{s'} P(s'|s,a) Q^{\pi^*}(s',a') \right\} \tag{12}$$

Where π^* is the optimal policy mapping sequence of states to action, Q^{π^*} is the optimal Q-function of the optimal policy.

In Q-learning, the Q-function is implemented as a table of states and actions pair, and then the values are updated iteratively as the agent accumulates knowledge. A linear approximation function for updating the weight can be sufficient if the simple environment to work with is relatively small. However, if the action space is of high dimension, the number of transactions to store gets more complex, then the use of a non-linear approximation approach such as deep neural networks becomes an option.

Deep neural network function-approximation with respect to its weights θ, is referred to as a deep Q-network. The weights θ_{is} are used to approximate the value function across the whole state-action space. The interaction data (s, a, r, s') are stored in a priority experience replay buffer (PER). The classifier agent will then randomly sample a mini-batch from the PER and perform stochastic gradient descent on the Q-network by minimizing a loss function.

$$L_i(\theta_i) = \sum_{(s,a,r,\, s') \in (PER)} \; (y_i - Q(s,a,s'))^2 \tag{13}$$

Where $y_i = \begin{cases} r & when \;\; failure \; is \; true \\ & \vdots \\ r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) & when \;\; failure \; is \; false \end{cases}$

Differentiating the loss function (equation 13) with respect to the weights θ_i we get

$$\nabla\theta_i L_i(\theta_i) = \sum_{(s,a,r,s',) \in PER} \; [y_i - Q(s,a;\theta_i) \nabla\theta_i Q(s,a;\theta_i)] \tag{14}$$

A Q(λ)-learning [43] is used to improve the algorithm learning process. In the process, SARSA learning is combined with eligibility trace [53] and incorporated into Q-learning to give a more general method that learns efficiently using time-series data. The eligibility trace considers a temporal history of the transaction (s,a,r), since we are using function approximation instead of a Q lookup table to estimate Q-values, a trace is considered for each component of the weight θ. The update is done as follows.

$$Q_{t+1} = Q_t(s,a) + \alpha\Delta_t e_t(s,a;\theta) \tag{15}$$

$$\theta_i = \theta_{i-1} + \alpha\Delta e_i \tag{16}$$

Where $\Delta_i = y_i - Q(s,a;\theta_i)$ is the SARSA error, and $e_i = \frac{\gamma\lambda e_i + \Delta Q(s,a;\theta_i)}{\Delta\theta_i}$ is the eligibility value.

II. Deep Q -learning with Prioritized Experience Replay (PER): In the normal Q-learning or DQN, the max operator uses the same value for both action selection and action evaluation[35]. This is likely to result in the selections that lead to over-optimistic estimation. Hado et al.[35] proposed double deep reinforcement learning (DDQN). The DDQN is designed to reduce over-estimation by decomposing the max operator in the target network into action selection and action evaluation. DDQN reduces the problem of over-estimation using two value functions by randomly assigning each experience to update one of the two value functions. That there are two sets of weight θ *and* θ' for every update, one set of weights is used to determine its value.

$$Q^d(s,a) = r_t + \gamma Q\left(s', \max_{a'} s,a;\theta\right);\theta' \tag{17}$$

Similarly, SARSA learning is a stochastic way of using the value of the action elected by an agent in the next step instead of using max as in Q-learning.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{18}$$

Therefore, double deep SARSA can be derived by substituting equations (15) in (17) to get

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[\Delta_t e_t(s,a;\theta);\theta'] \tag{19}$$

The use of experience replay memory to store observed transactions provides capabilities for reinforcement learning agent-classifier to remember past transaction experiences [56]. In the normal experience replay approach, the transactions are from time-to-time uniformly sampled from the buffer to update the network without considering any significance of the weight θ with respect to the policy π^*(in the DQN method, the policy is obtained implicitly by calculating a $Q_\theta(s,a)$ function, where the parameter θ measures the goodness of the given state-action with respect to policy). However, in a prioritized experience replay (PER) approach, the algorithm weighs the samples so that "important" ones are drawn more frequently for training [57]. The important samples are then played more frequently, which neglects the problem with strong correlations between consecutive samples. This technique improves the performance of the algorithm. Therefore, we adopted PER with double deep SARSA learning and DDQN learning as a building block for our proposed framework for predicting rare failures in the aircraft maintenance system, as seen in (algorithm 1) and (algorithm 2).

Algorithm 1: Double Deep SARSA- Learning

Input: Training Data $\boldsymbol{D} = \left\{\left[\left(\boldsymbol{x_{1,1}}, \boldsymbol{x_{1,2}} \ldots \boldsymbol{x_{1,n}}\right), \left(\boldsymbol{y_1}\right)\right], \ldots, \left[\left(\left(\boldsymbol{x_{m,1}}, \boldsymbol{x_{m,2}} \ldots \boldsymbol{x_{m,n}}\right), \left(\boldsymbol{y_n}\right)\right)\right]\right\}$

(Episode Number k, step-size n, replay period K, Size N and exponents α, β *and* budget T, PER =H)

Initialize replay memory H (H=$\phi, \Delta = 0, p_1 = 1$)

Initialize action-value Function Q with random weight θ, e= 0

Initialize Environment ε (observe s_0, *and choose* $A_o \sim \pi_\theta(s_0)$)

For k=1 to K do

Training data **d**

Initialize state $s_0 = x_0$

For t=1 to T do

Observe (s_t, R_t, γ_t), $a_t = \pi_\theta(s_t)$

Store transaction (s_t,a_t, γ_t, s'_t) in H with maximal priority $p_t = \overset{max}{i} < tp_i$

IF $t \equiv 0\ mod\ k\ then$

For j=1 to k do

Sample transaction $j \sim p(j) = \frac{p_j^{\propto}}{\sum_i p_i^{\propto}}$

Compute importance: sampling weight $\theta_j = \frac{\left(N.p(j)\right)^{-\beta}}{\underset{i}{max}\theta_i}$

Set $y_i = \begin{cases} r_j, & label_j = True \\ r_j + \alpha\left[\Delta_j\, e_j\,(s, a; \theta); \theta'\right], & and\ label_j = False \end{cases}$

Perform gradient descent on L(θ) w.r.t θ:

$$L_i(\theta_j) = \sum_{(s,a,r,s') \in H} \left(y_j - Q\left(s_j, a_j, s'_j;\ \theta_j\right); \theta_j'\right)^2$$

Update the transaction priority $p_j \leftarrow \left|\Delta_j\right|$

Accumulate weight change and traces.

End For loop

Update weights θ_i

Copy weight into the target network $Q_{Target} \leftarrow \theta$

End IF

Choose Action $A_t \sim \pi_\theta(s_t)$

End For loop

If window size = w, break

End For Loop

Algorithm 2: Double Deep Q-Network

Input: Training Data $\mathbf{D} = \left\{\left[\left(\boldsymbol{x_{1,1}}, \boldsymbol{x_{1,2}} \ldots \boldsymbol{x_{1,n}}\right), \left(\boldsymbol{y_1}\right)\right], \ldots, \left[\left(\left(\boldsymbol{x_{m,1}}, \boldsymbol{x_{m,2}} \ldots \boldsymbol{x_{m,n}}\right), \left(\boldsymbol{y_n}\right)\right)\right]\right\}$

(Episode Number k, step-size n, replay period K, Size N and exponents α, β *and* budget T, PER =H)

Initialize replay memory H (H=$\phi, \Delta = 0, p_1 = 1$)

Initialize action-value Function Q with random weight θ, e= 0

Initialize Environment ε (observe s_0, *and choose* $A_o \sim \pi_\theta(s_0)$)

For k=1 to K do

Training data **d**

Initialize state $s_0 = x_0$

For t=1 to T do

Observe (s_t, R_t, γ_t), $a_t = \pi_\theta(s_t)$

Store transaction $(s_t, a_t, \gamma_t, s'_t)$ in H with maximal priority $p_t = \overset{max}{i} < tp_i$

IF $t \equiv 0 \; mod \; k \; then$

For j=1 to k do

Sample transaction $j \sim p(j) = \frac{p_j^{\propto}}{\sum_i p_i^{\propto}}$

Compute importance: sampling weight $\theta_j = \frac{\left(N.p(j)\right)^{-\beta}}{\underset{i}{max}\theta_i}$

Set $y_i = \begin{cases} r_j, & label_j = True \\ r_j + \gamma \; \overset{max}{a} Q\left(s_{j+1}, a'; \theta\right), & and \; label_j = False \end{cases}$

Perform gradient descent on L(θ) w.r.t θ:

$$L_i(\theta_j) = \sum_{(s,a,r,s') \in H} \left(y_j - Q\left(s_j, a_j, s'_j; \theta_j\right); \theta_j'\right)^2$$

Update the transaction priority $p_j \leftarrow |\Delta_j|$

Accumulate weight change and traces.

End For loop

Update weights θ_i

Copy weight into the target network $Q_{Target} \leftarrow \theta$

End IF

Choose Action $A_t \sim \pi_\theta(s_t)$

End For loop

If window size = w, break

End For Loop

4. Case Study

The application of this novel technique is validated using a real-world ACMS dataset. The dataset is obtained from a fleet of aircraft. There are two families of aircraft in the long-range (A330) and the short aisle aircraft (A320). The first data is the data generated from the central maintenance system (log-based ACMS data), and the second data is the record of maintenance activities. The datasets are obtained from a fleet of long-range (A330) aircraft and A320 families. According to families, aircraft grouping is necessary because the data generated differ in properties and structure. The designation routes were different for each family; some were mainly used for long-distance routes, while some were primarily used for short distances. From the A330 aircraft family, the total number of failure/warning messages after pre-processing is 389902, and the A320 family has a total of 890120.

The main objective is to develop a predictive model to predict failure resulting in aircraft's unplanned repairs or components' replacement. Therefore, we choose target components, identified by Functional Item Number (FIN). The representation of these components is extremely rare. The basic idea is to detect both the extreme minority class samples and the majority class samples correctly during model classification. In each family, we target three components or functional items that are replaced due to unscheduled maintenance and study their failure behaviours. The behavioural patterns are then used to build a predictive model to predict their replacement. Data from the year 2011 to 2016 is used for training and testing (80% used for training and 20% used for testing), while the reaming from 2016 to 2018 is used for validation. The datasets are obtained from a fleet of long-range (A330) aircraft and A320 families. According to families, aircraft grouping is necessary because the data generated differ in properties and structure. The designation routes were different for each family; some were mainly used for long-distance routes, while some were primarily used for short distances.

EVENT_DATE	TAIL_NUMBE	FIN_REMOVALS	ATA	SOURCE	FAILURE MESSAGE
01/10/2016 08:23	CS-TOA		362215	BMC2	ENG2 PYLON LOOP INOP
02/10/2016 20:04	CS-TOA		316322	DMC1	DU ND CAPT (3WK1)
03/10/2016 05:02	CS-TOA		316322	DMC1	DU ND CAPT (3WK1)
03/10/2016 23:29	CS-TOA		316322	DMC1	DU ND CAPT (3WK1)
04/10/2016 09:53	CS-TOA		240000	FWS	POWER SUPPLY INTERRUPT
04/10/2016 09:53	CS-TOA		3150	FWS	FWS SDAC 2 FAULT
04/10/2016 09:53	CS-TOA		315534	FWS	SDAC2(1WV2)
04/10/2016 09:53	CS-TOA		3600		MAINTENANCE STATUS BMC 2
04/10/2016 09:53	CS-TOA		362215	BMC2	ENG2 PYLON LOOP INOP
04/10/2016 16:22	CS-TOA		212300	VC	GALY LAV DUCT CLOGGED
04/10/2016 16:22	CS-TOA		2128		MAINTENANCE STATUS CRG VENT
04/10/2016 16:22	CS-TOA		233234	CIDS2	PRAM (10RX)/ DIR2 (102RH)
04/10/2016 16:22	CS-TOA		237346	CIDS1	DEU A (200RH34)
04/10/2016 16:22	CS-TOA		307000	CIDS1	HEATR 119/ WIPCU AFT (200DW)

Figure 4. An example of a real ACMS

A proper and exhaustive analysis the ACMS data was carried out with close interaction with the domain expert to identify relationships among attributes and the "decision" variables of interest, cause-effect of failure. The ACMS data does not contain any description accompanying it, so our initial task was to understand the data characteristic and subsequently identify pre-processing and modelling requirements. As seen in Figure 4, columns (variables) available in the ACMS dataset are:

1. Event date: Date the failure message occurs

2. Aircraft Tail Number: Uniquely identified aircraft in the fleet data

3. FIN Removals: Identify the components removals

4. Failure Source: This Shows the sub-system that the failure message belongs to

5. Failure Message: Show the description of the failure message which relates to the physics of each components.

6. Leg of occurrence: Indicates the flight where the failure message occurs

7. TSI(FH): It shows time Since the installation of the component replaced

8. CSI (FC) : Cycles Since Installation (cycles)

9. Date Install (DT_INST): This shows the date the component/LRU was installed

10. DT_REM: Component/LRU Removal Date

11. RAZAO_REMO: Reason for Removal either as Scheduled or Unscheduled.

12. SIT: Situation at Removal either Serviceable or Unserviceable

13. Flight Phase: This shows the exact flight phase when the failure message was generated (e.g. take-off, cruise, and landing)

14. Departure Airport: Show the take-off airport

15. Arrival Airport: Destination Airport

16. Flight Number: show the flight number assigned to the particular aircraft. The dataset has a data imbalance problem, which the proposed technique seeks to address.

Among the several possible components identified by their Functional Identification Numbers (FIN), This thesis focused on the ones having a higher economic impact in operation according to the airline that provided the dataset.

Table 2. Selected Component to be considered in this study

Selected Components	
A330	A320
4000KS – Electronic Control Unit/ Electronic Engine Unit	**11HB** - Flow control valve
4001HA/4000HA – High-Pressure Bleed Valve	**10HQ** - Avionics equipment ventilation computer
5RV1- Satellite Data unit	**1TX1** - Air traffic service
438HC -- Trim Air Valve	**8HB** - Flow control valve 2

The analysis of maintenance records and considering the ACMS aircraft family shows that the total number of failure/warning messages after pre-processing for A330 is approximately 389902 in about 4023 flight legs. The A320 family has a total of roughly 890120 failure messages in about 10874 flight legs. The information indicated that the occurrence of unscheduled replacement for the targeted components occurs on averagely two to three times every thousand flights.

The data was split into training and testing divided into 70/30 (from January 2001 to September 2016) and validation data from October 2016 to April 2018 (without known label). A time-series window of 30 flight legs was used as a trajectory length. The choice is done based on domain expert advice. The leg size value is considered to be the memory size of the ACMS fault messages recording process. For this reason, it has been decided to analyse groups of no more than 30 consecutive legs. Meaning that 30 past flights are considered to identify predictive patterns.

4.1 Experiment

An experiment is set up to investigate the application of different deep reinforcement learning (DRL) architectures for the extreme rare failure prediction problem. The transformed DRL framework's implementation is based on the proposed DDSARSA and DDQN for extremely rare failure prediction. The implementation is based on the following.

I. DQN (Baseline): This is a normal deep Q-Network that uses a neural network to approximate a state-value function in a Q-learning framework. The baseline uses a standard experience replay memory.

II. DDQN+PER: In this implementation, we use the proposed double deep Q-learning with Prioritized Experience Replay memory to predict rare event failures in aircraft predictive maintenance modelling. The aim is to investigate the effectiveness of using DDQN+PER (see algorithm 2) by evaluating the effect of the overestimation problem and efficiency of the model in handling the extreme imbalanced data.

III. DDSARSA+PER: In this implementation, we use a DDSARSA with Prioritized Experience Replay memory to predict rare event failure in aircraft predictive maintenance modelling. The aim is to investigate the effectiveness of using double deep SARSA learning with PER (see algorithm 1) by evaluating the effects of PER and eligibility trace during learning and the model's efficiency in handling the extreme imbalanced problem.

IV. To investigate the proposed deep reinforcement learning approach's performance compared to other existing rare failure prediction methods. Three methods were considered: the Cost-Sensitive method [31], SMOTE with random forest [58], and the deep learning AE-BGRU [59]. The cost-sensitive method is an existing technique that modifies the loss function in Long Short Term Memory (LSTM) networks. The algorithm responds favourably to both classes during learning. The method is designed to handle rare failure prediction in time series datasets as implemented in previous work [31]. SMOTE+RF is a technique that balances the dataset using the Synthetic Minority Oversampling Technique (SMOTE) before presenting it as an input to the learning algorithm (Random Forests). The method is designed to handle extreme imbalanced classification problems [58]. AE-BGRU [59] is a strategy for predicting rare failure that uses a rescaled loss function in a hybrid deep network architecture known as the auto-encoder bidirectional gated recurrent network (AE-BGRU) model.

4.2 Description of the network architectures

There are two core approaches to data-driven maintenance, each geared towards different connected capabilities of aircraft or components. The network architecture consists of convolutional layers (CNN) and long-short team memory (LSTM) layers, which enhances the learning of the temporal dependency in the sequential data. A dense layer is also used to minimize the effect of overfitting, as seen in Table 3. The number of hidden layers and architecture design differ depending on the aircraft family dataset structure. For instance, an A320 aircraft will not transmit the same operational data level as the A320; hence, each dataset's learning strategies are different.

Table 3 Deep Network Network Architecture

Values	**Layers**
	Sequential
filters =32, kernel =3, activation = ReLU	Convolution 2D
	MaxPooling
Unit =32, dropout =0.2, activation = ReLU	LSTM
Unit = 1, activation =sigmoid	Dense (Fully-connected)

Where ReLU returns X if the value is positive else, it returns zero. Max-pooling is added after the convolutional layer reduces the feature map that is generated by the convolution operation. Max-pooling also helps in selecting only important information, which removes weak activation information hence avoiding overfitting problems. LSTM layer is added to correlated information from the past with current combined with the convolutional layer helps to learn better correlations between variables. The dense layer, also referred to as fully connected, is added as the last layer it is used to make the final decision based on the input from the LSTM layer.

4.3 Performance Metrics for the Models Validation

The problem is considered as an imbalance learning approach. Therefore, we use performance metrics relevant to evaluating rare event prediction. The algorithm is evaluated using the ACMS data. The data is split into training and testing divided into 70/30 (from January 2001 to September 2016) and validation data from October 2016 to April 2018 (without known label).

The performance metrics used are built from a definition presented by David [60]. Most machine learning and statistics best practise guides indicate a preference to partition available data as follows:

I	Use for the initial development of the model(s) say A, B and C
J	Used for establishing optimum model: A or B or C (when to stop training)
K	Used as final validation on previously unknown data of selected model

The assumption is that I, J and K each contain representative data of the whole population and are independent and identically distributed with adequate coverage in each partition.

As seen in Figure 5 for classification models, confusion matrices are often used to measure effectiveness in the validation of models. In this study, component failures are considered a positive class, while non-failure is considered a negative class.

Definition of evaluation formulae used in this study using Figure 5.

Patterns with component Failure = Positives

Patterns without component Failures = Negatives

True Positives (tp) = patterns with components failures that have been classified as a failure.

True Negatives (tn) = patterns without component failure who have been classified as non-failure

False Positives (fp) = patterns with components failure who have been classified as non-failure

False Negatives (fn) = patterns without components failure who have been classified as failures

Common metrics extracted from these are:

The true-positive rate (TPR), also known as Sensitivity, measures the proportion of components with failure who have been classified as component failures.

$$TPR/sensitivity = tp/(tp+fn) \quad (20)$$

The false-negative rate (TNR), also known as Specificity, measures the proportion of components without failure that has been classified as non-failure components.

$$TNR/Specificity = tn/(tn+fp) \quad (21)$$

The false-negative (FNR) measures the proportion of patterns without components failure who have been classified as failures.

$$FNR = fn/(tp+fn) \quad (22)$$

False Positive rate (FPR) measures the proportion of patterns with components failure that has been classified as non-failure.

$$FPR = fp/(fp+tn) \quad (23)$$

A false-positive arises when an example (pattern) from the minority class is misclassified as an example from the majority class. False-negative is less serious than false-positive when patterns with component failures are considered positives (minority class) and patterns without component failures

are considered negatives. The term "false-positive" is used in this study to describe misclassifying a malfunctioning component as "healthy," which is particularly dangerous because it could cause equipment damage. Similarly, a false negative involves misclassifying a working component as faulty; as a result, the extra cost of maintenance checks may increase.

The performance measurements are often used to compare several models using a ROC plot [61] (see Figure 5). In the case of failure prediction of an aircraft component, sensitivity is the model's ability to correctly predict failure, leading to component replacement (probability of positive prediction given that the failure results in component replacement). Model specificity relates to the model's ability to correctly predict non-failure, resulting in no replacement (probability of negative prediction given that no failure occurs).

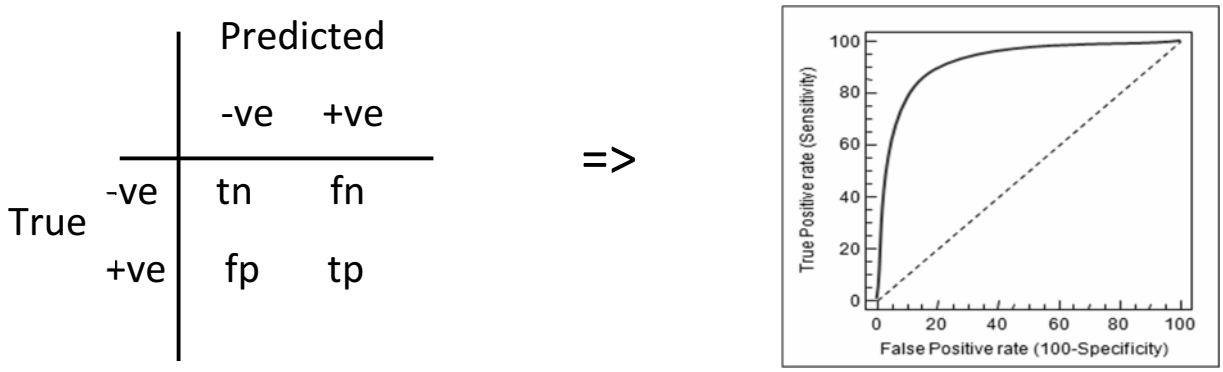


Figure 5. Confussion matrix and ROC curve

A very simple metric to measure classification performance is accuracy:

$$\text{Accuracy} = (tp + tn)/n \tag{24}$$

Accuracy is the ratio of correct predictions to the total number of samples in the dataset. However, this metric can be misleading in extreme imbalanced classes, as high metric values don't show the true prediction capability for the minority class. An accuracy of 99% can be achieved, but this still represents poor prediction capability of the class of interest. In such cases, accuracy could be misleading as one could predict the dominant class most of the time and still achieve a relatively high overall accuracy with correspondingly low precision or recall for other classes. To understand the current model predictive ability, the following metrics are used.

$$\text{Precision} = tp/(tp + fp) \tag{25}$$

a measure of how well the true events are classified.

$$\text{Recall} = tp/(tp + fn) \quad (26)$$

which is the fraction of instances of a class that were correctly predicted. Geometric mean (G-mean):

$$\text{G-mean} = \sqrt{(precision * recall)} \quad (27)$$

is commonly reported as a metric that measures the balance between classification performances on both the majority and minority classes. It measures the harmonic mean average between precision and recall.

The metrics presented above show that accuracy and specificity are influenced by population size and can, therefore, distort the measure of classifier performance. Consequently, it is worth considering sensitivity, recall, precision, FPR, FNR and G-mean values as these aren't distorted by population size, particularly where highly imbalanced datasets are involved.

5. Results and Discussion

Each algorithm was run five times for each target event with the same hyperparameter for 200 epochs using five random seeds. The Q-function is approximated using deep neural networks.

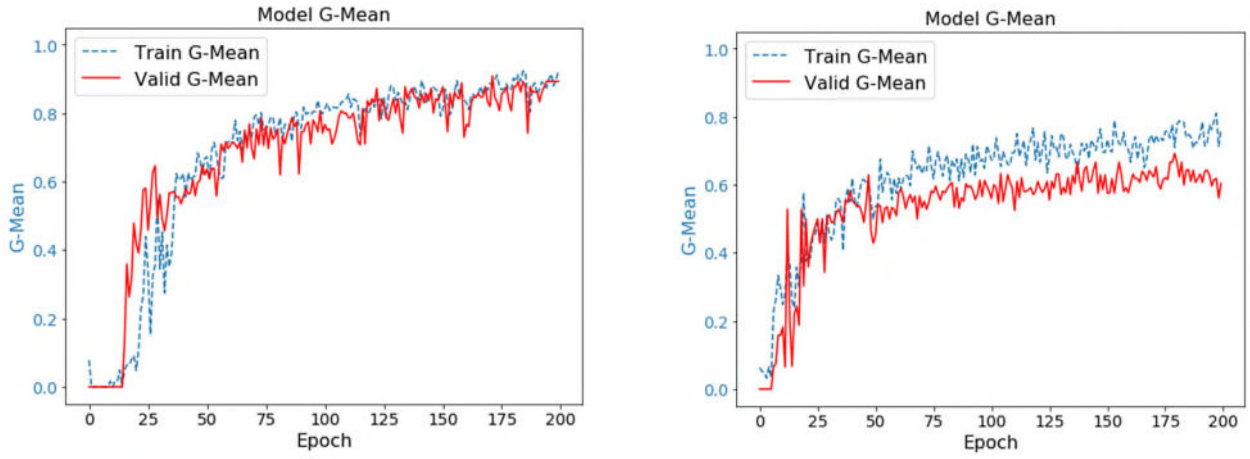
5.1 Results

The first investigation performed was to verify the applicability and effectiveness of using DDSARSA+PER and the Deep Q-network for rare failure prediction. As seen in Table 4, these investigations' results are compared with a baseline method DQN (deep Q-Network). The result indicates that DDSARSA+PER and DDQN+PER can effectively be applied for rare failure prediction or data Imbalanced classification. It can generally be observed that the two novel implementations show significant improvement in terms of model performance. Although there is a delay in the training time, there is a significant reduction in both the FPR and FNR, which is very important for aircraft maintenance applications. The impact of eligibility trace positively impacts the new algorithms by reinforcing entire sequences of actions from a single experience, contributing to the improved performance in the proposed algorithms.

Table 4. Shows DDSARSA learning with PER and DDQN with PER for rare failure prediction.

Aircraft ACMS Dataset											
			DQN (Baseline)			DDQN+PER			**DDSARSA+PER**		
	LRU	ρ	G-mean	FPR	FNR	G-mean	FPR	FNR	**G-mean**	**FPR**	**FNR**
A330-Family	4000KS	0.0043	0.77	0.0023	0.021	0.85	0.0015	0.004	**0.94**	**0.00023**	**0.0100**
	4001HA	0.0047	0.79	0.0021	0.018	0.86	0.0013	0.003	**0.97**	**0.00023**	**0.0800**
	5RV1	0.0044	0.78	0.0020	0.017	0.84	0.0017	0.004	**0.95**	**0.00012**	**0.0111**
A320 Family	11HB	0.0028	0.71	0.0025	0.023	0.82	0.0014	0.0011	**0.90**	**0.00009**	**0.0000**
	10HQ	0.0031	0.75	0.0021	0.019	0.84	0.0011	0.0125	**0.93**	**0.00009**	**0.0000**
	1TX1	0.0064	0.80	0.0020	0.018	0.89	0.0010	0.011	**0.98**	**0.00002**	**0.0001**

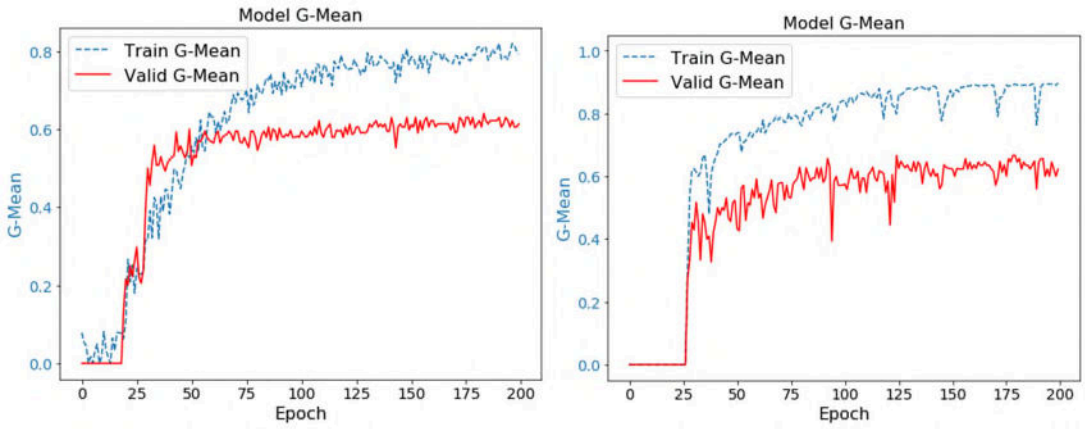
**** The bold results show that the DDSARSA+PER method outperforms the baseline methodology and the DDQN+PER strategy in terms of G-mean, FPR, and FNR.**



(a) *(b)*

Figure 6. Summary of the model performance in terms of G-Mean (data from A330 aircraft family, for component 4001HA)

(a) double Deep SARSA (b) Double Deep Q_Network with prioritized experience replay memory model



(a) *(b)*

Figure 7. Summary of model performance in terms of G-Mean data (A320 aircraft family for the component 1TX1)

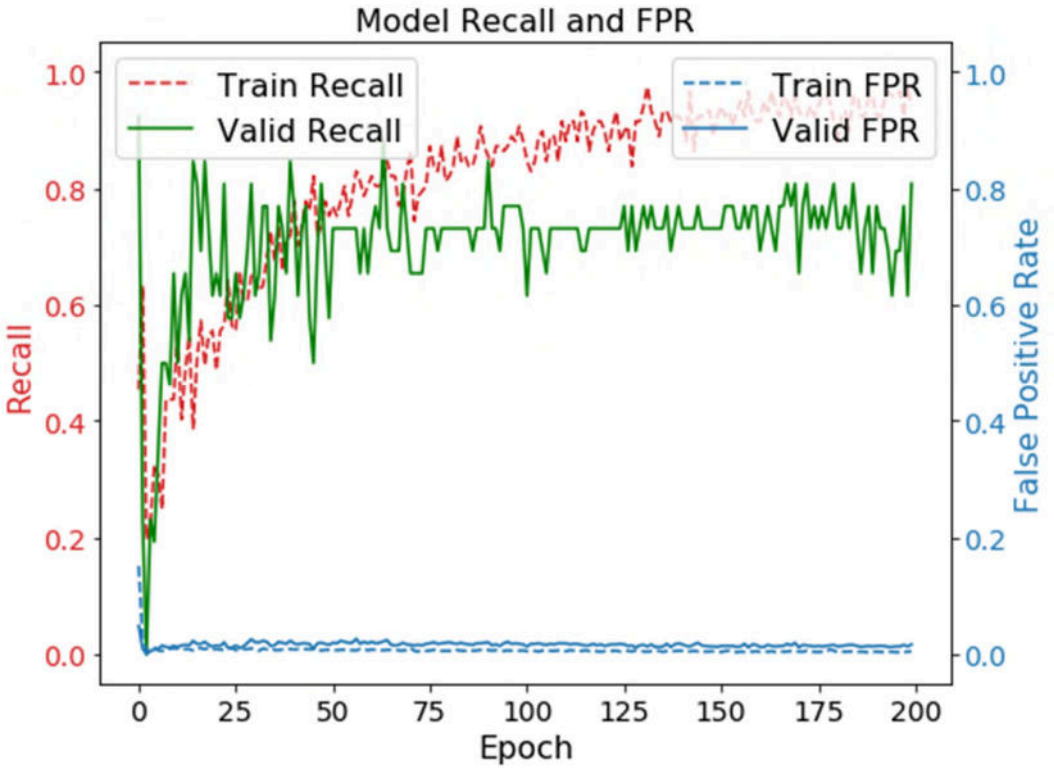
(a) double Deep SARSA (b) Double Deep Q_Network with prioritized experience replay memory model

Figures 6 and 7 show the classifier-agent performance over the validation dataset for both A330 and A320 aircraft, respectively. The model is trained for up to 200 epochs, rewarded with the parameter ρ as seen in Table 4, and a learning rate of 0.01. Figure 6(a) shows the performance of the DDSARSA+PER model, and it can be observed that the agent learns slowly between 0- 25 epochs for validation. After 25 epochs, the performance increases steadily and normalizes at 0.7g-mean for validation. Similar performance is seen in Figure 6(b), which shows the performance of DDQN+PER, the model learns slowly up to 15 epochs, and the performance increases steadily, achieving 0.65g-mean for validation. The model's performance on A320 aircraft is seen in Figures 7(a) and 7(b); as observed, the DDSARSA+PER model shows better G-mean performance than DDQN+PER.

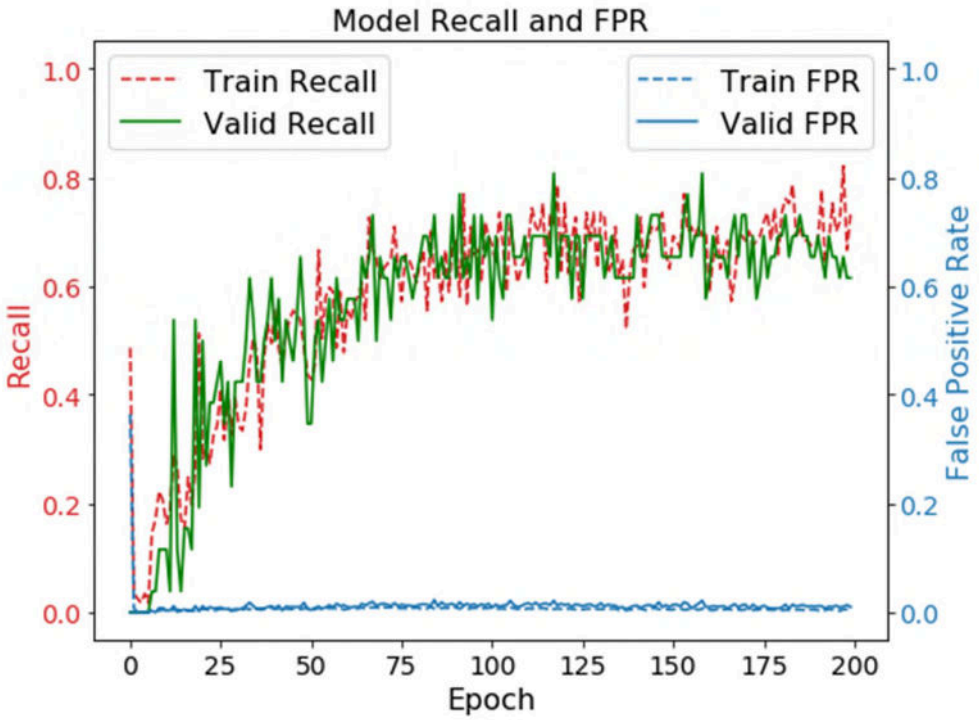
It is important to note that the choice of hyper-parameter λ and the imbalance ratio ρ significantly impact the model's overall performance because they can cause the agent to learn a sub-optimal policy. When the value of λ is large, the model converges quicker at the G-mean's expense, and when the value is small, the model converges slower with better performance. DDSARSA+PER only gives better performance at a certain value of λ based on the structure and complexity (i.e. the length of the sequence pattern for each failure) on the dataset in question. Adjusting and keeping the reward function's parameter lambda (λ) static impacts the algorithm's performance. Therefore, to improve learning on the proposed approach, we performed a grid search in each training phase to dynamically adjust the hyper-parameters reward function (λ) based on the use-case imbalance ratio ρ. The parameter lambda (λ) is define in the range [0,1]

5.1.1 Model sensitivity analysis

Figures 8 and 9 illustrate model performance results based on the recall and FPR for training and validation on the A330 and A320 aircraft families. From each dataset family, one component was picked. A330's 4001HA (high-pressure bleed valve) and A320's 1TX1 (air traffic control unit).



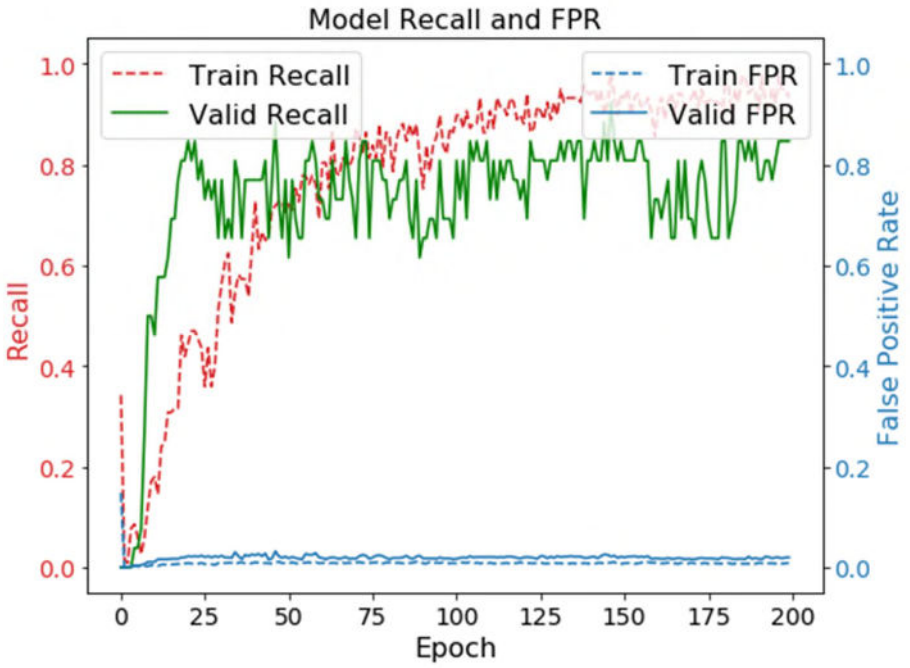
(a)



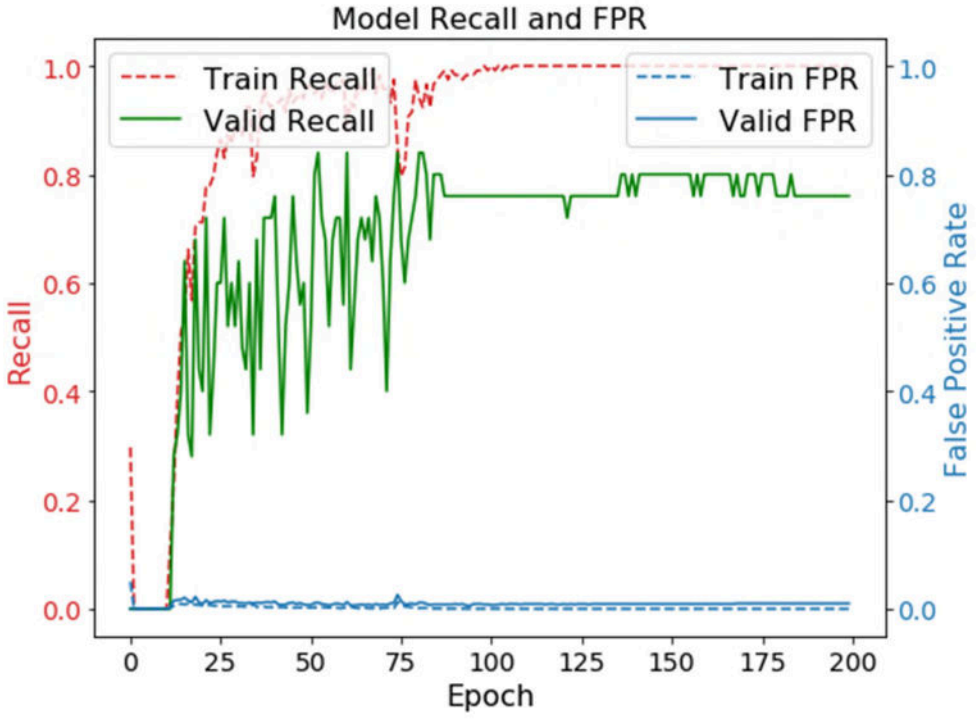
(b)

Figure 8. Summary of the model performance in terms of false-positive rate (data from A330 aircraft family for the component 4001HA) (a) double Deep SARSA (b) Double Deep Q_Network with prioritized experience replay memory model.

The result of training the DDSARSA+PER algorithm using the A330 dataset is shown in Figure 8(a). the result indicates that it takes roughly 150 epochs to reach 0.85 recall for the validation data, with a consistent FPR of about 0.00023 (see Table 4) during the validation period. As observed, DDSARSA shows a more robust training capability than the DDQN+PER in 8(b) with an FPR of 0.0013 (see table 4) and a validation recall of 0.76.



(a)



(b)

Figure 9. Summary of the model performance in terms of false-positive rate (data from A320 aircraft family for the component 1TX1) (a) double Deep SARSA (b) Double Deep Q_Network with prioritized experience replay memory model.

The model validation for both the DDSARSA+PER and DDQN+PER methods utilising the A320 aircraft family dataset is shown in Figure 9. Figure 9(a) demonstrates how the DDSARSA+PER model, with a validation score of 0.85 recall and an FPR of 0.00002, converges quicker after 100 epochs and shows a better training capability than the DDQN+PER with a validation score of 0.80

recall and an FPR of 0.011. It is worth noting that the models in both implementations (DDSARSA and DDQN) had a low false-positive rate for all test situations. However, DDSARSA+PER, on the other hand, offers the advantage of faster convergence and robustness in handling extremely imbalanced problems, as shown in g-mean and FPR scores.

Furthermore, false alarms in equipment predictive maintenance might result in increased maintenance expenditures due to unnecessary checks. It may also lower the level of trust in the equipment prognostics system. As a result, the goal is to keep FPR and FNR as low as possible while maintaining a solid G-mean. The proposed approach (DDSARSA+PER) and other existing imbalance learning methods (Cost-sensitive ensemble and random forest with SMOTE) are compared in terms of False Positive Rate (FPR), as shown in Figure 10.

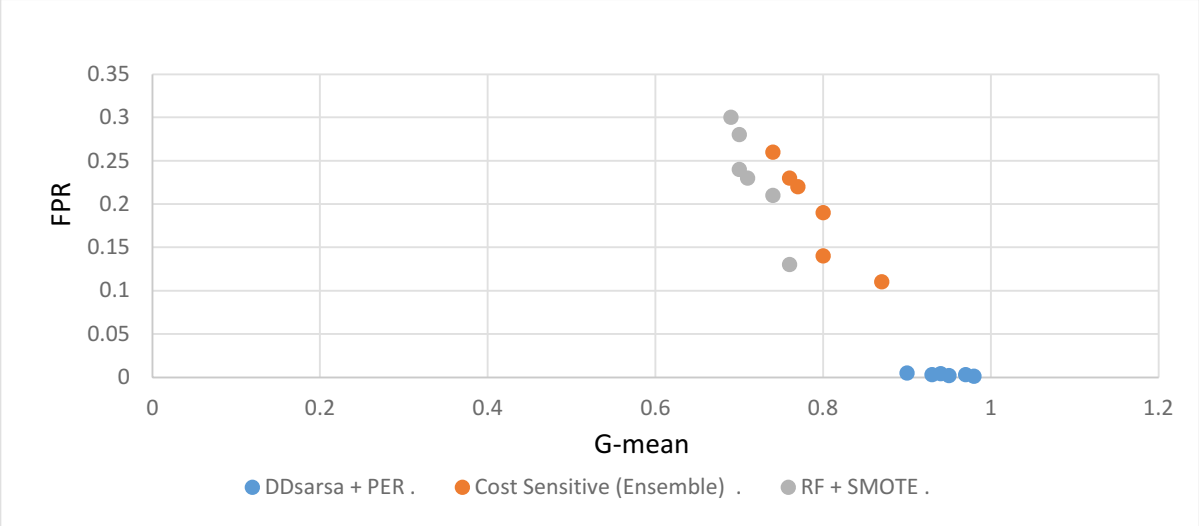


Figure 10. Performance Analysis in terms of False Positive Rate (FPR) between the proposed algorithm other existing state-of-the-earth imbalance learning methods

In terms of FPR, the proposed method (DDSARSA+PER) outperformed both the algorithm level (cost-sensitive learning) [31] and the data level (Random Forest with Synthetic oversampling) [58] methods. Figure 10 shows that the FPR for the DDSARSA+PER is less than 0.001 in all situations studied, while cost-sensitive approaches have FPRs ranging from 0.11 to 0.26, and SMOTE+RF methods have FPRs ranging from 0.13 to 0.3. In terms of G-mean and FPR, the overall result demonstrates that the Cost-Sensitive approach and the SMOTE+RF method perform similarly on both datasets (A330 and A320 aircraft).

5.1.2 Model Validation in Predicting Failure Within a given Range

Further research was conducted to establish the model's ability to anticipate aircraft component failure within the specified time frame, such as the ability to predict a number of flights ahead of failure. It is critical to make predictions within a realistic time frame, not too far ahead of the failure

point to prevent wasting resources, and not too near to the failure point to allow enough time to plan maintenance. As a result, ten to two flights prior to a failure point is considered a reasonable period for raising an alert.

Figure 11 depicts a graphical picture of the timeframe that leads to failure. Point zero denotes the actual failure point, whereas points less than zero (negative) denote flights prior to the failure and points larger than zero (positive) denote flights following the failure. The following requirements were considered when using the DDSARSA+PER model and ACMS testing data (representing data from previous flights without labels) to make predictions: any failure alert that arrives earlier than -10 is considered too early, and any failure alert that arrives later than -2 flights before the actual failure point (zero) is considered too late prediction.

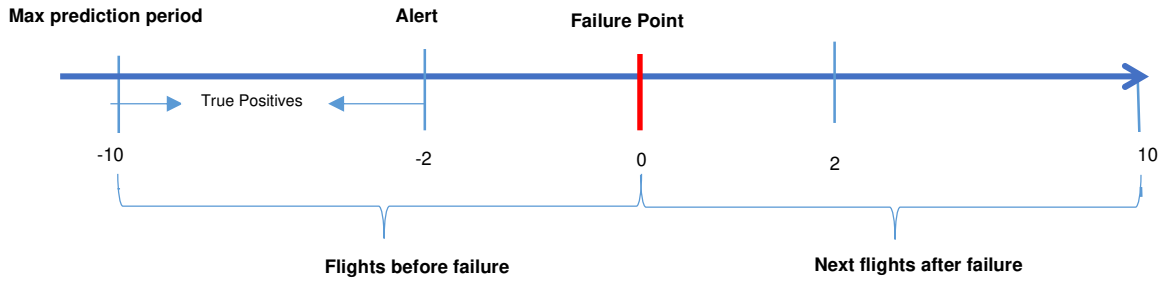


Figure 11. Flight cycles before (indicated with nagive sign) and after failure

The following components are picked from the A330 family (4000KS,4001HA, 5RV1) and the A320 family (11HB, 10HQ,1TX1) to test the model's effectiveness at the fleet level.

The predicted results are displayed in Figure 12. Each point represents the difference between the time of actual maintenance action and its predicted time (prediction residual). The residual error between two and ten (shown by the red lines) are true positives; that is, the model predicted component replacement within the desired range. Those above ten indicate the prediction came too early, and those below two indicate the model predicted maintenance too late. Residual error at point zero naturally represents the points for which maintenance and prediction were simultaneous, and negative values show a very late prediction.

It can be seen that the majority of the failure alerts for the component 4000KS (electronic engine unit) are within the target range. Only three alerts came too early, and one alert was predicted very late. For the component 4001HA (pressure regulating valve), three alerts are predicted too early, and two are predicted late, with two at precisely on the failure leg (zero), and two were predicted very late (below zero). Likewise, for the component 5RV1 (satellite data unit), the model predicted most of the failures within the target range. Similar performance is seen in the A320 aircraft family, with

the model predicting a majority of failures for 11HB (the flow control valve), 10HQ (avionics equipment ventilation computer), and the 1TX1 (air traffic service unit) within the target range.

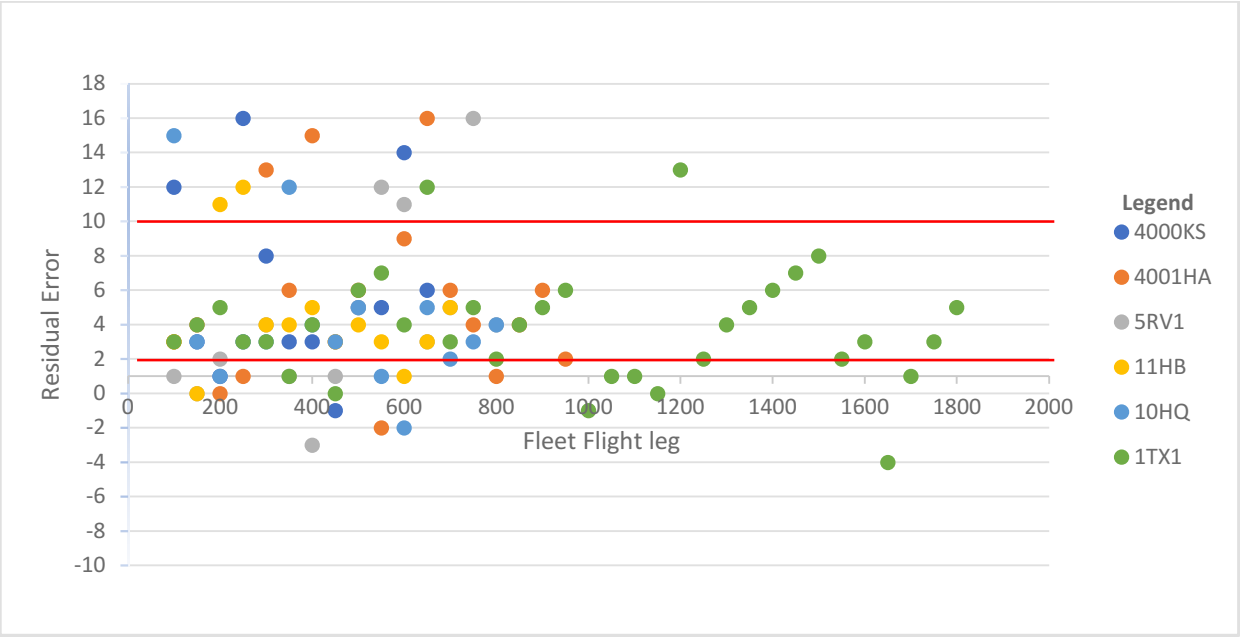
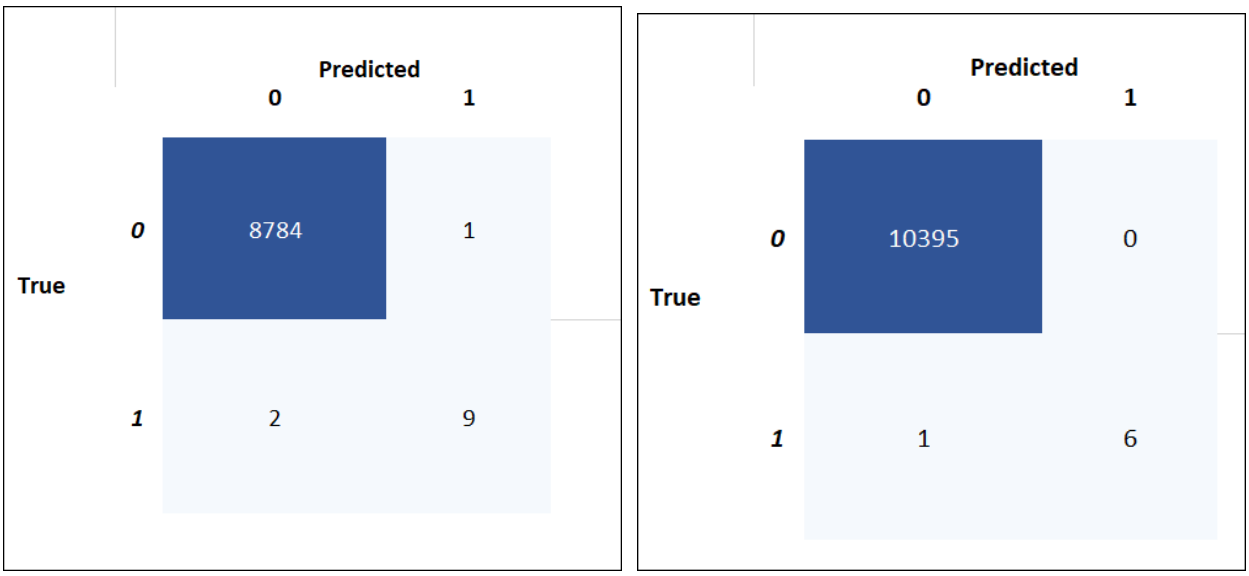


Figure 12. Validation of Proposed Model against actual maintenance record

In conclusion, based on the prediction score in Figure 12, the proposed DDSARSA+PER model can forecast approximately 90% of aircraft component replacements within a specific range, i.e. not more than ten flights and not fewer than two flights to failure.

The number of failure cases classified is shown in Figures 13 (a) and 13 (b). The proposed model's confusion matrix was created using one component from the A330 and A320 datasets. As shown in Figure 13(a), the DDSARSA+PER model successfully predicted 9 out of 11 unplanned electronic engine unit failures (4000KS from the A330 dataset). Figure 13(b) shows the model predicted 6 out of 7 flow control valve failures (11HB from the A320 dataset).



(a) (b)

Figure 13 (a) 4000KS - Electronic Control Unit/ Electronic Engine Unit (b)11HB - Flow Contol Valve

5.1.3 Model of Comparative Analysis

The comparative analysis between the proposed method (DDSARSA+PER) and previous studied imbalance learning methods (cost-sensitive and SMOTE) and the autoencoder with bidirectional gated recurrent unit (AE-BGRU) network [59]. It can be observed that despite the extreme imbalance ratio in all the cases considered for both the A330 and A320 datasets, the proposed method performs much better in terms of G-mean and FPR. For example, looking at Table 5, considering 4000KS with the lowest imbalance of 0.0043, it can be observed that the G-mean for DDSARSA+PER is 0. 94 while that of cost-sensitive is 0.74, SMOTE+RF is 0.70 and 0.66. A similar performance is seen for other components, with a higher imbalance ratio compared to 4000KS. This clearly shows performance supremacy for the DDSARSA+PER model in predicting rare failure.

Table 5. The performance of the proposed reinforcement learning approach with other existing rare failure prediction methods
**** The bold results show that the DDSARSA+PER method outperforms the other similar strategy in terms of G-mean, FPR, and FNR.**

			DDSARSA + PER			Cost-Sensitive (LSTM)			SMOTE+RF			AE-BGRU		
	Component	ρ	**G-mean**	**FPR**	**FNR**	G-mean	FPR	FNR	G-mean	FPR	FNR	G-Mean	FPR	FNR
A330 Family	4000KS	0.0043	**0.94**	**0.00023**	**0.0100**	0.74	0.026	0.103	0.70	0.024	0.022	0.66	0.0083	0.3800
	4001HA	0.0047	**0.97**	**0.00023**	**0.0800**	0.80	0.014	0.111	0.74	0.021	0.024	0.63	0.0013	0.4615
	5RV1	0.0044	**0.95**	**0.00012**	**0.0111**	0.76	0.023	0.106	0.71	0.023	0.022	0.65	0.0008	0.4000
(A320) Family	11HB	0.0028	**0.90**	**0.00009**	**0.0000**	0.77	0.022	0.101	0.69	0.030	0.023	0.62	0.0019	0.5555
	10HQ	0.0031	**0.93**	**0.00009**	**0.0000**	0.80	0.019	0.104	0.70	0.028	0.022	0.65	0.0028	0.5454
	1TX1	0.0064	**0.98**	**0.00002**	**0.0001**	0.87	0.011	0.101	0.76	0.013	0.017	0.81	0.0002	0.2352
Average prediction score			**0.95**	**0.00013**	**0.0168**	0.79	0.0192	0.1043	0.71	0.023	0.021	0.67	0.0025	0.4296

The performance improvement in the deep reinforcement learning implementation, especially the DDSARSA+PER model, comes from a number of different factors such as the reward function, which optimize future rewards, in contrast to a machine learning model that predicts the probability of future outcomes (classification using an ensemble method and the synthetic minority oversampling techniques with random forest). Secondly, the use of PER, which, instead of uniformly sampling transactions from replay memory, employs a prioritized approach. This also entails the replay of the important transactions more frequently and hence learns more effectively.

5.2 Discussion

The main aim of this study is to investigate the applicability of deep reinforcement learning techniques for training an extremely rare failure predictive model instead of the widely used machine learning or deep learning methods for slightly imbalanced datasets. Two algorithms (the DDSARSA and the DDQN) are designed and implemented. The implementation results show that the application of deep reinforcement learning for extremely rare failure prediction is viable, and the constructed algorithm shows superior performance as compared with baseline DQN. Also, it was observed that the proposed DDSARSA+PER algorithm shows better learning as compared to DDQN+PER. Then DDSARSA+PER was compared with existing imbalanced learning methods, and performance was evaluated based on G-mean and false-negative rates. As indicated in Table 5, the average prediction rate for all six components was calculated, in comparison to the cost-sensitive LSTM approach with 0.79 g-mean and 0.019 FPR, SMOTE+RF with 0.71 g-mean and 0.022 FPR, and autoencoder with a

bidirectional gated unit network with 0.67 g-mean and 0.0026 FPR, the proposed DDSARSA showed superior performance with an overall 0.95 g-mean score and an average of 0.0005 FPR.

Calculating the percentage increase in G-mean scores from the highest g-mean, the Cost-Sensitive (LSTM) model, which is 0.79, to the highest g-mean, DDSARSA+PER, which is 0.95, the result shows that DDSARSA exhibits a 20.3% g-mean improvement. In addition, when computing the percentage drop, the suggested model reduces FPR by roughly 97.3684 % by using the lowest FPR, which is 0.019 to 0.0005.

The overall observation shows that the Cost-Sensitive method and the oversampling (SMOTE+RF) method perform relatively the same on both datasets (A330 and A320 aircraft) in terms of G-mean and FPR. What accounts for the significant performance improvement in DDSARSA are basically the combination of the convolutions in deep neural networks which enhance learning relationships between variables in the dataset, the reward function which helps to counter bias during model training and the use of prioritised experience replay memory, instead of uniformly sampling transactions from replay memory, employs a prioritised approach; this also entails replaying the important transactions more frequently, optimising the learning process. Also, DRL uses a reward function to optimise future rewards, in contrast to a machine learning (regression or classification) model that predicts the probability of future outcomes. Therefore, it can be concluded that deep reinforcement learning methods are ideally best for imbalanced classification problems because of their learning mechanism and specific learning environment and reward function. The PER and eligibility trace also contributed to the performance impact. The impact of eligibility trace positively impacts the new algorithms by reinforcing entire sequences of actions from a single experience, contributing to the improved performance in the proposed algorithms.

The impact of FNR, that is, the proportion of "healthy" components classified as failures in equipment' predictive maintenance, can result in higher maintenance costs due to unnecessary checks. Also, FPR - the proportion of faulty components classified as non-faulty or when the model fails to predict failure can result in equipment damage or huge loss. A high FPR score or FNR score might potentially lower the level of trust in the equipment prognostics system. As a result, the goal is to bring both FNR and FPR down to an acceptable level. This implies the model should accurately identify fewer false alarms, lowering total operational costs and increasing vehicle availability and reliability. As shown in Figure 10 in comparing the proposed model to existing approaches, the proposed DDSARASA+PER model shows a lower false-negative rate. The usage of a double deep neural network is the main disadvantage of DDSARSA+PER, which increases training time but can be compensated for by a high detection rate. This study will impact research towards mitigating unscheduled maintenance for systematic schedule maintenance.

6. Conclusion

In this study, a novel technique for predicting extremely rare failure is proposed and implemented. The new technique is based on a deep reinforcement learning approach. Two algorithms are constructed, the double deep Q-Network with prioritized experience replay memory and the double deep state-action-reward-state-action with prioritized experience replay memory. The effectiveness of the new approach is validated using a real-world aircraft central maintenance log-based dataset. The result shows that the application of deep reinforcement learning for extremely rare failure prediction is viable. It also indicates that the proposed double deep state-action-reward-state-action with prioritized experience replay memory model can effectively predict component failure in both the A330 and A320 aircraft families with low false-positive and false-negative rates. The result means that unscheduled maintenance can be reduced in the aircraft fleet at the same time decreasing the cost of maintenance operations.

The work can be extended by carrying out further experimentation to determine the impact of high imbalanced on other deep reinforcement learning. Parameters such as changing the network architecture, an additional variable can be introduced into the deep neural network to keep track of the physical state and check for inconsistency with the physical laws to improve accuracy. Also, future work can consider enhancing performance optimization using other deep reinforcement learning algorithms. An ablation study will be carried out to assess the impact of eligibility trace and prioritise experience replay memory individually. More aircraft data sources - such as quick access recorder (QAR) Data, Performance Reports (PR), and Maintenance Tech Logs data can be integrated into the analysis.

Acknowledgement

The authors would like to acknowledge the Integrated Vehicle Health Management Centre (IVHM), Cranfield University, and the first author would like to thank PTDF Nigeria for sponsoring his study.

Reference

[1] Korvesis P. Machine Learning for Predictive Maintenance in Aviation. 2017. https://doi.org/theses.fr/2017SACLX093.

[2] Martinez C, Perrin G, Ramasso E, Rombaut M. A deep reinforcement learning approach for early classification of time series. Eur Signal Process Conf 2018;2018-Septe:2030–4. https://doi.org/10.23919/EUSIPCO.2018.8553544.

[3] Lin E, Chen Q, Qi X. Deep reinforcement learning for imbalanced classification. Appl Intell 2020. https://doi.org/10.1007/s10489-020-01637-z.

[4] Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N. A survey on addressing high-class imbalance in big data. J Big Data 2018;5. https://doi.org/10.1186/s40537-018-0151-6.

[5] Ran Y, Zhou X, Lin P, Wen Y, Deng R. A Survey of Predictive Maintenance: Systems, Purposes and Approaches 2019;XX:1–36.

[6] Patel H, Singh Rajput D, Thippa Reddy G, Iwendi C, Kashif Bashir A, Jo O. A review on classification of imbalanced data for wireless sensor networks. Int J Distrib Sens Networks 2020;16. https://doi.org/10.1177/1550147720916404.

[7] Burnaev E. Rare Failure Prediction via Event Matching for Aerospace Applications. 2019 3rd Int Conf Circuits, Syst Simulation, ICCSS 2019 2019:214–20. https://doi.org/10.1109/CIRSYSSIM.2019.8935598.

[8] François-lavet V, Henderson P, Islam R, Bellemare MG, François-lavet V, Pineau J, et al. An Introduction to Deep Reinforcement Learning. (arXiv:1811.12560v1 [cs.LG]) http://arxiv.org/abs/1811.12560. Found Trends Mach Learn 2018;II:1–140. https://doi.org/10.1561/2200000071.Vincent.

[9] Çinar ZM, Nuhu AA, Zeeshan Q, Korhan O, Asmael M, Safaei B. Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. Sustain 2020;12. https://doi.org/10.3390/su12198211.

[10] Daigle MJ, Goebel K. Model-based prognostics with concurrent damage progression processes. IEEE Trans Syst Man, Cybern Part ASystems Humans 2013;43:535–46. https://doi.org/10.1109/TSMCA.2012.2207109.

[11] Wu D, Jennings C, Terpenny J, Gao RX, Kumara S. A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. J Manuf Sci Eng 2017;139:071018. https://doi.org/10.1115/1.4036350.

[12] Schwabacher M. A Survey of Data-Driven Prognostics. Infotech@Aerospace 2005. https://doi.org/10.2514/6.2005-7002.

[13] Woodward PW, Castillo E. Extreme Value Theory in Engineering. vol. 42. 1993. https://doi.org/10.2307/2348127.

[14] Falk M. Multivariate Extreme Value Theory and D-Norms. 2019. https://doi.org/10.1007/978-3-030-03819-9.

[15] Kamarujjaman, Maitra M, Chakraborty S. A novel decision-based adaptive feedback median filter for high density impulse noise suppression. Multimed Tools Appl 2020. https://doi.org/10.1007/s11042-020-09473-6.

[16] Rydman M. Application of the Peaks-Over-Threshold Method on Insurance Data. Uppsala Univ UUDM Proj Rep 2018;32:1–21.

[17] Murphy KP. Machine Learning A Probabilistic Perspective. 2012. https://doi.org/10.1007/978-94-011-3532-0_2.

[18] Bzdok D, Altman N, Krzywinski M. Points of Significance: Statistics versus machine learning. Nat Methods 2018;15:233–4. https://doi.org/10.1038/nmeth.4642.

[19] Laptev N, Yosinski J, Erran Li L, Smyl S, Li EL, Smyl S. Time-series Extreme Event Forecasting with Neural Networks at Uber. Int Conf Mach Learn - Time Ser Work 2017:1–5.

[20] Allen RJ, Valeriani C, Rein Ten Wolde P. Forward flux sampling for rare event simulations. J Phys Condens Matter 2009;21. https://doi.org/10.1088/0953-8984/21/46/463102.

[21] Berberidis C, Angelis L, Vlahavas I. Inter-transaction association rules mining for rare events prediction. Proc 3rd Hell Conf … 2004.

[22] Sammouri W, Côme E, Oukhellou L, Aknin P, Fonlladosa C-E. Floating train data systems for preventive maintenance: A data mining approach. Proc. 2013 Int. Conf. Ind. Eng. Syst. Manag. IEEE - IESM 2013, 2013.

[23] Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: A brief survey. IEEE Signal Process Mag 2017;34:26–38. https://doi.org/10.1109/MSP.2017.2743240.

[24] Moniz N, Monteiro H. No Free Lunch in imbalanced learning. Knowledge-Based Syst 2021;227:107222. https://doi.org/10.1016/j.knosys.2021.107222.

[25] Fernández Alberto, Garcia Salvador, Galar Mikel, Prati Ronaldo, Krawczyk Bartosz HF. Learning From Imbalanced Data Sets. 2018. https://doi.org/https://link.springer.com/content/pdf/10.1007%2F978-3-319-98074-4.pdf.

[26] Chawla N V., Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. J Artif Intell Res 2002;16:321–57. https://doi.org/10.1613/jair.953.

[27] Lusa L, Others. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 2013;14:106. https://doi.org/10.1186/1471-2105-14-106.

[28] Hu Y, Guo D, Fan Z, Dong C, Huang Q, Xie S, et al. An Improved Algorithm for Imbalanced Data and Small Sample Size Classification. J Data Anal Inf Process 2015;03:27–33. https://doi.org/10.4236/jdaip.2015.33004.

[29] Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G. Learning from class-imbalanced data: Review of methods and applications. Expert Syst Appl 2017;73:220–39. https://doi.org/10.1016/j.eswa.2016.12.035.

[30] Wu Z, Lin W, Ji Y. An Integrated Ensemble Learning Model for Imbalanced Fault Diagnostics and Prognostics. IEEE Access 2018;6:8394–402. https://doi.org/10.1109/ACCESS.2018.2807121.

[31] David Dangut M, Skaf Z, Jennions I. Rescaled-LSTM for Predicting Aircraft Component Replacement Under Imbalanced Dataset Constraint. 2020 Adv. Sci. Eng. Technol. Int. Conf., IEEE; 2020, p. 1–9. https://doi.org/10.1109/ASET48392.2020.9118253.

[32] Qi D, Majda AJ. Using machine learning to predict extreme events in complex systems. Proc Natl Acad Sci U S A 2020;117:52–9. https://doi.org/10.1073/pnas.1917285117.

[33] Johnson JM, Khoshgoftaar TM. Survey on deep learning with class imbalance. J Big Data 2019;6. https://doi.org/10.1186/s40537-019-0192-5.

[34] Keneshloo Y, Shi T, Ramakrishnan N, Reddy CK. Deep Reinforcement Learning for Sequence-to-Sequence Models. IEEE Trans Neural Networks Learn Syst 2019:1–21. https://doi.org/10.1109/tnnls.2019.2929141.

[35] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-Learning. 30th AAAI Conf Artif Intell AAAI 2016 2016:2094–100.

[36] Luong NC, Hoang DT, Gong S, Niyato D, Wang P, Liang YC, et al. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. IEEE Commun Surv Tutorials 2019;21:3133–74. https://doi.org/10.1109/COMST.2019.2916583.

[37] Li C, Qiu M, Li C. Reinforcement Learning for Cybersecurity. Reinf Learn Cyber-Physical Syst 2019:155–68. https://doi.org/10.1201/9781351006620-7.

[38] Mosavi A, Ghamisi P, Faghan Y, Duan P. Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics 2020:1–43. https://doi.org/10.20944/preprints202003.0309.v1.

[39] Chakraborty S. Capturing Financial markets to apply Deep Reinforcement Learning 2019:1–17.

[40] Gijsbrechts J, Boute RN, Van Mieghem JA, Zhang D. Can Deep Reinforcement Learning Improve Inventory Management? Performance and Implementation of Dual Sourcing-Mode Problems. SSRN Electron J 2019:1–26. https://doi.org/10.2139/ssrn.3302881.

[41] Jonsson A. Deep Reinforcement Learning in Medicine. Kidney Dis 2019;5:18–22. https://doi.org/10.1159/000492670.

[42] Waschneck B, Reichstaller A, Belzner L, Altenmüller T, Bauernhansl T, Knapp A, et al. Optimization of global production scheduling with deep reinforcement learning. Procedia CIRP 2018;72:1264–9. https://doi.org/10.1016/j.procir.2018.03.212.

[43] Lee XY, Balu A, Stoecklein D, Ganapathysubramanian B, Sarkar S. A case study of deep reinforcement learning for engineering design: Application to microfluidic devices for flow sculpting. J Mech Des Trans ASME 2019;141:1–10. https://doi.org/10.1115/1.4044397.

[44] Knowles M, Baglee D, Wermter S. Reinforcement learning for scheduling of maintenance. Res Dev Intell Syst XXVII Inc Appl Innov Intel Sys XVIII - AI 2010, 30th SGAI Int Conf Innov Tech Appl Artif Intel 2011:409–22. https://doi.org/10.1007/978-0-85729-130-1-31.

[45] Rocchetta R, Bellani L, Compare M, Zio E, Patelli E. A reinforcement learning framework for optimal operation and maintenance of power grids. Appl Energy 2019;241:291–301. https://doi.org/10.1016/j.apenergy.2019.03.027.

[46] Zhang C, Gupta C, Farahat A, Ristovski K, Ghosh D. Equipment health indicator learning using deep reinforcement learning. Lect Notes Comput Sci (Including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 2019;11053 LNAI:488–504. https://doi.org/10.1007/978-3-030-10997-4_30.

[47] Wiering M, Schmidhuber J. Fast Online Q(λ). Mach Learn 1998;33:105–15. https://doi.org/10.1023/A:1007562800292.

[48] Martin M. Bellman equations and optimal policies. Learning 2011. https://doi.org/https://www.cs.upc.edu/~mmartin/Ag4-4x.pdf.

[49] Montague PR. Reinforcement Learning: An Introduction, by Sutton, R.S. and Barto, A.G. Trends Cogn Sci 1999;3:360. https://doi.org/10.1016/s1364-6613(99)01331-5.

[50] Rummery GA, Niranjan M. ON-LINE Q-LEARNING USING CONNECTINIST SYSTEMS. Cambridge, Engl Univ Cambridge, Dep Eng 1994;37:20.

[51] Bouneffouf D, Bouzeghoub A, Gançarski AL. Following the user's interests in mobile context-aware recommender systems: The hybrid-e-greedy algorithm. Proc - 26th IEEE Int

Conf Adv Inf Netw Appl Work WAINA 2012 2012:657–62. https://doi.org/10.1109/WAINA.2012.200.

[52] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518:529–33. https://doi.org/10.1038/nature14236.

[53] Mousavi SS, Schukat M, Mannion P. Applying Q(λ)-learning in Deep Reinforcement Learning to Play Atari Games. Ala 2017:1–6.

[54] Barron EN, Ishii H. The Bellman equation for minimizing the maximum cost. Nonlinear Anal 1989;13:1067–90. https://doi.org/10.1016/0362-546X(89)90096-5.

[55] Silver D. Markov decision processes. Adv Comput Vis Pattern Recognit 2015;54:199–216. https://doi.org/10.1007/978-1-4471-6699-3_11.

[56] Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. 4th Int Conf Learn Represent ICLR 2016 - Conf Track Proc 2016:1–21.

[57] Liu R, Zou J. The Effects of Memory Replay in Reinforcement Learning. 2018 56th Annu Allert Conf Commun Control Comput Allert 2018 2019:478–85. https://doi.org/10.1109/ALLERTON.2018.8636075.

[58] Dangut MD, Skaf Z, Jennions IK. An integrated machine learning model for aircraft components rare failure prognostics with log-based dataset. ISA Trans 2021;113:127–39. https://doi.org/10.1016/j.isatra.2020.05.001.

[59] Dangut MD, Skaf Z, Jennions IK. Rare Failure Prediction Using an Integrated Auto-encoder and Bidirectional Gated Recurrent Unit Network. IFAC-PapersOnLine 2020;53:276–82. https://doi.org/10.1016/j.ifacol.2020.11.045.

[60] Powers DMW. Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation. Int J Mach Learn Technol 21 (2011), Pp37-63 2007. https://doi.org/arXiv:2010.16061 [cs.LG].

[61] Roc B. Comparing Two ROC Curves – Independent Groups Design. NCSS, LLC 2021:1–26.

2022-02-08

Application of deep reinforcement learning for extremely rare failure prediction in aircraft maintenance

Dangut, Maren David

Elsevier

Dangut MD, Jennions IK, King S, Skaf Z. (2022) Application of deep reinforcement learning for extremely rare failure prediction in aircraft maintenance. Mechanical Systems and Signal Processing, Volume 171, May 2022, Article number 108873

https://doi.org/10.1016/j.ymssp.2022.108873