The Aeronautical Journal (2024), 1–20
doi:10.1017/aer.2024.23

REGULAR PAPER

Distributed digital twins for health monitoring: resource constrained aero-engine fleet management

A. Hartwell[1], F. Montana[1], W. Jacobs[1], V. Kadirkamanathan[1], N. Ameri[2] and A. R. Mills[1]

[1]Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK
[2]Rolls-Royce Plc, Bristol, UK
Corresponding author: Andrew R. Mills; Email: a.r.mills@sheffield.ac.uk

Received: 27 October 2023; **Revised:** 17 February 2024; **Accepted:** 26 February 2024

Keywords: digital twin; health monitoring; machine learning; fault diagnosis

Abstract

Sensed data from high-value engineering systems is being increasingly exploited to optimise their operation and maintenance. In aerospace, returning all measured data to a central repository is prohibitively expensive, often causing useful, high-value data to be discarded. The ability to detect, prioritise and return useful data on asset and in real-time is vital to move toward more sustainable maintenance activities.

We present a data-driven solution for on-line detection and prioritisation of anomalous data that is centrally processed and used to update individualised digital twins (DT) distributed onto remote machines. The DT is embodied as a convolutional neural network (CNN) optimised for real-time execution on a resource constrained gas turbine monitoring computer. The CNN generates a state prediction with uncertainty, which is used as a metric to select informative data for transfer to a remote fleet monitoring system. The received data is screened for faults before updating the weights on the CNN, which are synchronised between real and virtual asset.

Results show the successful detection of a known in-flight engine fault and the collection of data related to high novelty pre-cursor events that were previously unrecognised. We demonstrate that data related to novel operation are also identified for transfer to the fleet monitoring system, allowing model improvement by retraining. In addition to these industrial dataset results, reproducible examples are provided for a public domain NASA dataset.

The data prioritisation solution is capable of running in real-time on production-standard low-power embedded hardware and is deployed on the Rolls-Royce Pearl 15 engines.

Nomenclature

Acronyms

CNN	convolutional neural network
CPU	central processor unit
DT	digital twin
N1	fan speed [%]
GB	gigabyte
GTE	gas turbine engine
LReLU	leaky rectified linear units
MAE	mean absolute error
MB	megabyte
OS	operating system
P30	pressure near compressor exit [psi]
RAdam	rectified Adam optimizer algorithm
T30	temperature near compressor exit [K]
TGT	turbine gas temperature near low pressure turbine inlet [K]

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

Symbols

α_n	natural log of model prediction standard deviation
B	nodel training batch size
$\tilde{d}$	prediction distance score
e_n	difference between model prediction and signal
g	neuron activation function
K	convolution kernel object
$\mathcal{L}$	likelihood function
$\mathcal{LL}$	log-Likelihood function
N	length of model data window
s_j	neuron's temporally convolved signal
S	number of signals used by model instance
T	number of data samples used by model instance
x_n, y_n	model input data array and output scalar
X, y	model input and output dataset
z_j^l	neuron output
σ_n	standard deviation of model prediction
μ_n	mean of model prediction

Subscripts

n	nth sample of model dataset
j	jth neuron of a neural network layer
l	lth neural network layer

1.0 Introduction

Equipment monitoring systems rely on rich data streams to make operational decisions on asset usage and maintenance, but the data can overload communications and require computing that is both commercially and environmentally expensive [1].

The motivating example of aerospace gas turbine engines (GTE) provides context and motivation for the development of the methodology. Aerospace GTEs exhibit the problem attributes of complex machines that must be monitored and maintained in order to ensure reliable operation over long time periods [2]. GTEs are highly non-linear, dynamic and subject to many internal and external disturbances, and their behaviour is measured by large raw data streams (>1GB/hour) from an array of sensors. It is desirable to use the data to inform fleet-wide condition-based maintenance decisions [3], however communication bandwidth and on-board storage limitations mean that it is not possible to return all data to ground for comparative fleet analysis. It is, therefore, necessary to reduce the volume of data returned by a factor of 1,000 times (i.e. reduce data file volume to low MBs for the communication means available in aerospace systems). Any data prioritisation process must function in real-time using the existing limited computational resources lest data become unavailable before it can be stored.

For equipment health monitoring, data indicating deviation from design intent is of paramount importance, whereas ideal nominal behaviour is known to propulsion engineers. Thus we can define important data as that which is novel relative to design expectation. The digital twin paradigm [4, 5] provides a useful basis for novelty detection, where an accurate model simulation of an individual 'as-operated' machine is used to provide a simulation of a correctly functioning asset.

A common digital twin implementation utilises physics-based models that run in parallel to the real machine. Physical models have advantages, particularly in terms of explicability and their ability to extrapolate (when designed correctly). However, they are costly to develop, difficult to tune to an individual asset, typically bespoke, and are computationally expensive to run [6, 7]. An alternative paradigm is data-driven modelling, where a flexible model structure is trained on historical data to approximate the measured outputs of the machine. Data-driven models are often poor in terms of explicability and ability to extrapolate beyond the training set – but are more accurate in prediction for individual assets with a reduced computational burden once trained. It is typical that a digital twin will need to adapt to a changing physical plant as it ages, is maintained or changes its operational conditions. This requires a feedback mechanism of data to allow its evolution while also supporting decision making on when to perform preventative maintenance.

In our proposed framework, digital twins are distributed on both the asset, for access to full data bandwidth, and on-ground to benefit from fleet data and human decision makers. In this paradigm, we class any information returned from the machine as redundant if it matches that predictable by the digital twin. Conversely, any residual difference between data and model indicates value in acquiring the data for further investigation into potential changes in the machine and environment and for improving the digital twin model. Thus the effectiveness of detection of machine data novelty relative to its digital twin is the key determinant of the solution efficacy.

By detecting data that are novel relative to a model we can ensure that only data with information content is given attention and resources. It is not known how novelty will manifest ahead of time and therefore we must define normal in order to determine when anomalies occur. This viewpoint is also particularly useful in the gas turbine context because their high reliability means that the vast majority of available data are of nominal operation. Further, a prediction approach allows a proxy measurement for detection in the form of prediction error. This is useful because it provides an indicator of model performance even when there are limited data with faults available.

The difference between model prediction and measurement, termed the residual, has been widely researched as a fault detection technique. The difficulty in obtaining an accurate model for the plant led to significant early work [8] on the development of diagnostic methods with plant observers (using data measurements to adapt state estimates), and with parity equation approaches decoupling known disturbances from faults into model subspaces [9, 10]. These methods rely on analytical forms of the plant, faults and expected disturbances. The difficulties in constructing suitable models for complex machine behaviour (e.g. its non-linear nature and inter-unit variance) have led to these approaches being augmented with data-driven ‘tuning’ elements [11, 12] at the cost of increased complexity. Machine learning offers a more generic, lower cost and faster to market approach, as well as the potential to accurately capture modelled physics behaviours embedded in the data.

Other researchers have explored full or partial gas turbine anomaly detection (with special interest paid to anomaly detection in combustors) using the full gamut of machine learning techniques [13]. Deep learning techniques have been of particular relevance to the area in recent years [14–20] due to their success in other fields. These solutions are not suited to deployment on resource-limited hardware due to their high computational cost at run-time and/or their memory requirements.

Anomaly detection using limited hardware has also been explored, using remote processing [21], look-up tables [22] and model-based comparison [23]. These solutions, however, do not readily account for systems dynamics and hence are unsuitable for gas turbine system anomaly detection.

Non-linearity and high state dimensionality coupled with the variety of operating conditions and environmental factors presents a data coverage problem for anomaly detection. It is difficult to acquire enough data to well-defined normal for all possible conditions, environments and modes of operation. Therefore, we designed our models following Nix and Weigend [24], so that each model predicts a probability distribution characterised by a mean and confidence.

We propose a solution based on lightweight machine learning models running on the asset to continuously rank the utility of newly generated data at source and select only data of interest to store and transmit to synchronise with an on-ground digital twin. There is, therefore, a need for a novelty detection-based prioritisation method that is capable of running on-board under strict computational resource limitations. To address this need, in this paper we present a data-driven model and decision logic that is capable of real-time anomaly detection and prioritisation on embedded hardware.

Our research novelty is derived from a combination of a flexible modelling approach (allowing prediction of any input-output path within a machine), a confidence prediction for quantifying uncertainty

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

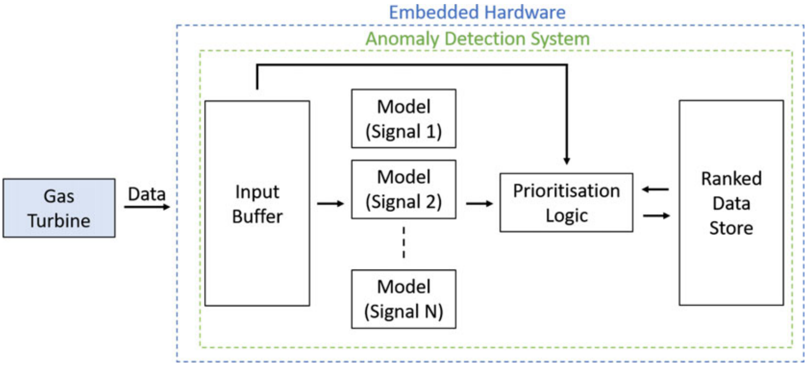

Figure 1. *On-board architecture. Data from the gas turbine is streamed into a temporary buffer before being run through the currently loaded models. Data with a higher novelty or lower confidence score replaces lower ranked items in the data store.*

in the models, and software that allows for real-time operation on embedded hardware. Further, our solution is especially flexible as it allows drop-in, and drop-out of models without retraining.

2.0 Methods

Novelty detection methodologies are used to determine the utility, expressed as a data priority, in returning data to the ground. To achieve this, a bespoke convolution neural network (CNN) has been designed to capture the dynamic behaviour of multi-input and non-linear physical machines. This model allows rapid update with new data and efficient deployment on embedded computing using model compression techniques. It acts as both the digital twin and, when deployed on the machine, plays a central role in the novelty detection mechanism. Other arrangements where the CNN is a surrogate to a physics-based digital twin, which is updated with novel data, are also equally possible where model explicability is paramount.

The confidence weighted model error, calculated on windows of time-series data, is used to provide two rankings. One ranking for the most novel data relative to the model and the second ranking based on the highest prediction uncertainty. Available communication bandwidth and media costs are used in decision logic to determine a maximum number of data windows for transmission to ground. Novel data may be linked to fault modes whereas high uncertainty can be associated with an operating regime outside of the training set. A fleet-level assessment of the returned data and the use of transfer learning strategies to update the CNN are discussed in Ref. [25].

2.1 Data prioritisation architecture

Data prioritisation is critical to determine the most informative, and hence potentially valuable, data to send over a limited communication channel to a centralised fleet management facility. Data are returned ordered on their novelty until a maximum data volume limit is reached.

The on-board architecture (see Fig. 1) allows real-time streaming of gas turbine data from sensors (signals) into a local *input buffer*. The buffer is used to feed data snapshots from a subset of signals of finite length to N *models*, where N is selected to provide at least one estimate of each signal needed for decision making. The model output is a novelty score indexed to the input data, which is passed to the *prioritisation logic*. The logic updates the *ranked data store* replacing lower novelty data snapshots with more novel data while avoiding exceeding storage limits.

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

Each model is designed to predict only a single output, multiple models with different input-output subsets are then required to run in parallel to provide wide coverage of machine states. For example, we may predict compressor pressure as a model output based on a short data window of past shaft speeds, fuel flow and compressor inlet gas temperature. This allows each model to be made lean and computationally efficient while simultaneously providing a simple way to scale to the computational resources available without retraining existing models. Training is performed using nominal data to make a prediction about a single signal given its input subset.

The model also predicts a confidence on its prediction based on the model input. Model inputs that are not well described by the training data, e.g. the specific engine operates on routes where high altitudes have not been flown, lead to a low confidence in prediction – thus allowing the distinction between model errors that arise from the unusual data of interest, and those from previously unseen operating conditions [24]. The model output forms an anomaly detector, which can be used to rank the novelty and uncertainty of data generated by the machine.

A storage buffer is filled with the non-overlapping data windows that generate high uncertainty or error when passed through the model. Once the buffer is filled, replacement of windows with higher metrics occurs as the live data stream is passed through the model. The data stored can be any sensor information from the machine and need not be limited to a model input-output subset. Data in the store may be accessed whenever a data link is available, for example, via satellite link when in-flight, or airport connectivity when landed. The store may be sized to match memory requirements of the hardware.

2.2 Data modelling

The anomaly detection scheme presented in the previous section is reliant on the existence of models that can map sensor observations of the input data to observations of output data. Consider a single input-output data set $\mathcal{D} = \{X, \mathbf{y}\}$, where

$$\mathbf{y} = [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^{N\times 1}, \tag{1}$$

$$X = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_N^T]^T \in \mathbb{R}^{N\times m}, \tag{2}$$

where N is the length of the data series and m is the number of sensor signals in the input data. The input-output mapping is approximated with a function $f(\cdot)$, such that the n'th element of the output can be modelled as

$$y_n = f(\mathbf{x}_n) + e_n, \tag{3}$$

where e_n is additive noise that is dependant on the input data.

The estimation of the distribution of the output data is central to the anomaly detection scheme, allowing the quantification of uncertainty on the estimate. Importantly, the uncertainty is a function of the input data $\mathbf{x}_n$ and quantifies the uncertainty in the model in different regions of the input space. The output distribution is described by a multivariate Gaussian with mean $\mu_n = \mu(\mathbf{x}_n)$ and standard deviation $\sigma_n = \sigma(\mathbf{x}_n)$ given by

$$\mathcal{N}\left(y_n|\mu_n, \sigma_n^2\right) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(y_n-\mu_n)^2}{2\sigma_n^2}}. \tag{4}$$

In this work, the input dependant mean and variance are estimated simultaneously using a deep neural network, the design, training and real-time implementation of which are discussed in the remainder of this section.

2.2.1 Neural network design

Recurrent neural networks are an often-favoured choice when dealing with time-series data, however their high computational and memory costs make them generally unsuitable to implement on current

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

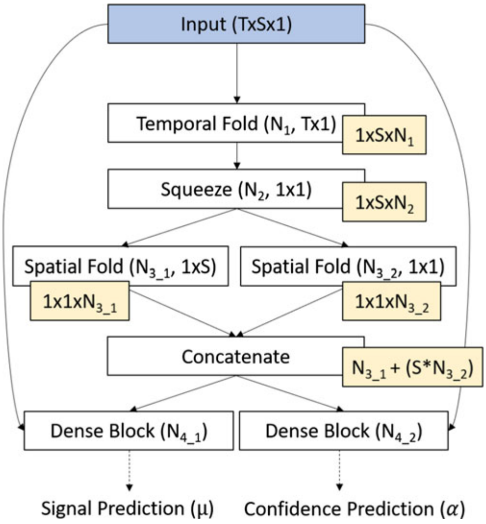

Figure 2. *Overview of the network architecture. Number of hidden units and stride size given in brackets, output size given in yellow. The design is flexible such that 'T' matches the number of incoming timesteps in each window and 'S' is the number of signals. The 'Dense Blocks' are three dense layers each followed by a dropout layer with rate = 0.5. The number of hidden units (N_X) in different layers may be adjusted based on the complexity of the input-output set and available computation.*

embedded hardware [26]. Therefore, we chose to use a CNN architecture in order to minimise the computational cost of forward passes and an ability to compress the models for efficient execution on embedded controllers [27].

The large number of sensors and the complexity of the gas turbine led to our decision to use a flexible network design that can be adapted to a variety of target input-output data set. Further we also engineered the design to be flexible in terms of the number of time-steps the network can take as an input. Typically this requires a large number of extra weights to scale the number of time-steps. Our design uses a constrained convolutional layer (the temporal fold) to minimise the computational impact of inputting extra time-steps. This is especially important as, while some input-output subsets are expected to have short time-lag relationships, others, e.g. temperatures, may have much longer time constants.

The design is flexible in terms of the number of inputs, which allows for deployment on new data by non-specialists; no redesign of the architecture is necessary to accommodate different input-output subsets or timescales. The network architecture takes advantage of skip connections to improve performance at minimal computational cost, dropout regularisation to reduce over-fitting [28], and leaky rectified linear units (LReLU) throughout. The conceptual architecture is shown in Fig. 2, which is described in more detail below.

Input Incoming data is treated as a 2D window with length T and width S, where T and S are time steps and the number of signals for a particular input-output subset respectively. This simplifies implementation and visualisation as it means the data can be treated in a similar manner to images.

Temporal fold The stage of the network is a 1D convolution in the time dimension, explicitly enforcing feature extraction from individual signals at the lowest level, here named the temporal fold. Denoting the convolution operator as $*$, the discrete 1D convolution operation is defined as

$$s_j^{[l]} = \left(\mathbf{z}_j^{[l-1]} * \mathbf{K}\right) = \sum_i \mathbf{z}_j^{[l-1]}(i) \odot \mathbf{K}(i) \tag{5}$$

where $\mathbf{z}_j^{[l-1]}$ is the output of the j'th neuron at the l'th layer of the network, $\mathbf{K}$ is a 1D kernel of appropriate size and $\odot$ is the element-wise product operator. The temporal fold takes the network input as input

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

$\mathbf{z}_j^{[0]}(i) = \mathbf{x}_{k-T:k}$ with $\mathbf{K}(i)$ equal to the ith column of $\mathbf{K}_{TF}$, the temporal fold kernel, with height T, width 1, and depth equal to the number of input signals. The output of the convolution can be seen as a weighted average over the past T data points for each input channel, where the weighting is defined by the elements of $\mathbf{K}_{TF}$. The output of the convolution is passed through a LReLU activation function defined as

$$z_j^{[l]} = g\left(s_j^{[l]}\right) \tag{6}$$

$$g\left(s_j^{[l]}\right) = \left(\begin{array}{ll} s_j^{[l]} & \text{if } s_j^{[l]} > 0, \\ 0.01 s_j^{[l]} & \text{otherwise} \end{array}\right. \tag{7}$$

Squeeze The next stage of the network is the squeeze layer. It consists of a 1 × 1 squeeze convolution which outputs to a 1 × S convolution and a 1 × 1 convolution, both described by Equation (5) with appropriately sized kernels and with LReLU activation functions described by Equation (7). This method is inspired by Squeezenet [29], and replaces the $c \times c$ convolution for a $1 \times S$ convolution in order to significantly reduce the number of parameters to be learned.

Spatial fold The two outputs of the squeeze layer are passed through a 1D convolution described by Equation (5), where $\mathbf{z}_j^{[l-1]}(i)$ indicates ith element of the row vector $\mathbf{z}_j^{[l-1]}$ and $\mathbf{K}(i)$ is the ith element of the spatial fold kernel $\mathbf{K}_{SF}$. The convolution is therefore performed in the spatial dimension (across the time convolved signals output at the previous layer). A LReLU activation function is again used. The layer determines cross-channel features and inter-filter features; these features are multi-signal transforms that contain increasingly concentrated information essential for novelty identification. The outputs of the convolutions are flattened and concatenated.

Dense layers – mean prediction: A skip connection is performed, where the output of the spatial fold is concatenated with the flattened original input. The concatenated outputs are then passed into a series of three dense layers described by

$$z_j^{[l]} = g\left(W z_j^{[l-1]} + \mathbf{b}\right) \tag{8}$$

where W and $\mathbf{b}$ are parameters to be learned and $g(.)$ is the LReLU activation function defined by Equation (7). A dropout layer with $rate = 0.5$ is placed after each dense layer to reduce over-fitting during training. The output is a single value that is a mean estimate of the output for the given network input.

Dense layers – confidence prediction: A further skip connection is performed where, again, the output of the spatial fold is concatenated with the flattened original input, but here the mean estimate is also concatenated. This is then passed through into a series of three dense layers and produces an estimate of the log output standard deviation.

2.2.2 Neural network optimisation

Optimisation of the model weights of a deep neural network requires an optimisation cost function. Considering the model described by Equation (4), it is simple to construct a likelihood function as

$$\mathcal{L} = \prod_{n=1}^{B} \mathcal{N}\left(y_n | \mu_n, \sigma_n^2\right), \tag{9}$$

where B is the batch size. Maximisation of the likelihood is equivalent to the minimisation of the negative log-likelihood

$$\mathcal{LL} = -\frac{1}{B} \sum\nolimits_{n=1}^{B} \ln\left(\frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2}\left(\frac{(y_n-\mu_n)^2}{\sigma_n^2}\right)}\right) \tag{10}$$

$$= \frac{1}{B} \sum\nolimits_{n=1}^{B} \left(\frac{1}{2}\left(\frac{(y_n-\mu_n)^2}{\sigma_n^2}\right) - \ln(\sigma_n) - \frac{1}{2} ln(2\pi)\right) \tag{11}$$

which is used as the cost function for optimisation [24].

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

Optimisation was performed using the rectified Adam (RAdam) algorithm [30, 31] combined with early stopping on a separate validation data split. The parameters of the rectified Adam algorithm were varied depending on the input-output subset based on experiments with the validation set. RAdam was selected based on its performance relative to a small pool of candidate optimisers.

The network outputs the mean, μ_n, and the confidence $\alpha_n = \ln(\sigma_n)$. The re-parameterisation enforces $\sigma_n > 0$ to ensure that the network output is always valid. σ_n is regenerated after the forward pass using $\sigma_n = e^{\alpha_n}$. The assumption of normally distributed residuals is reasonable, given that we expect random fluctuations from sensor readings.

This confidence estimation approach is based on the work of Nix and Weigend [24] and similar to more recent work e.g. bounding box estimation in computer vision [32]. Other approaches to uncertainty estimation in neural networks include using dropout at run-time [33], using ensembles as a prediction scatter and Bayesian neural network solutions. We utilise the explicit estimation approach since it allows us to enforce a Gaussian confidence estimation while retaining the greater flexibility of non-Bayesian neural networks and fast run-time of a single network solution, hence allowing on-board implementation.

2.2.3 Model and hyper-parameter selection

In order to ensure a neural network was an appropriate model choice, a number of simpler models were tested on real engine data as a benchmark. These models included lasso, ridge, elastic net linear regression and regression forests. Each was found to perform worse than the neural network solution and naturally lacked the ability to estimate confidence. Typically it was possible to train these models to work well at a local scale, performing the same or better than the neural network over small portions of the data, but considerably worse elsewhere indicating a tendency to over-fit to the training data and a lack of generalisation.

Network hyper-parameters were selected based on starting values from the literature followed by jittered grid search and manual tuning. Validation set (not test set) data was used to evaluate performance and inform selection. More information on the hyper-parameters are included in Appendix A.

2.2.4 Model comparison and data prioritisation

The system-level assessment of abnormality relies on being able to compare measurements to the outputs of any number of different models where each model may have a different output range. This is achieved by treating outputs of the network as a multivariate Gaussian and adopting the Mahalanobis distance [34]. Since each model is independent, such that the co-variance matrix is diagonal, the Mahalanobis distance reduces to standardised Euclidean distance. This leads to the consideration of the squared distance in Equation (12), referred to as the novelty score, that retains the intended functionality, but can be computed quickly in an online context.

$$\tilde{d}(y_n, \mu_n, \sigma_n) = \frac{(y_n - \mu_n)^2}{\sigma_n^2}, \qquad \sigma_n = e^{\alpha_n} \tag{12}$$

A gas turbine control system has existing data validation to detect measurement errors and marking signals as invalid. Models will not be executed in such sensor fault cases, and it is therefore important to have a diverse set of models which require different inputs to ensure availability of novelty assessment.

2.2.5 Real-time implementation

Deployment to embedded hardware was achieved via an application written in C++. The TensorFlow Lite [35] library was employed to ensure fast forward-passes of all models. This process transforms a trained flexible machine learning model into a limited inference only model with a lightweight compute engine. To maximise performance, no model specific optimisation (e.g. neither quantisation nor pruning [27, 36, 37]) has been applied since the memory footprint was within the computational constrains of our

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

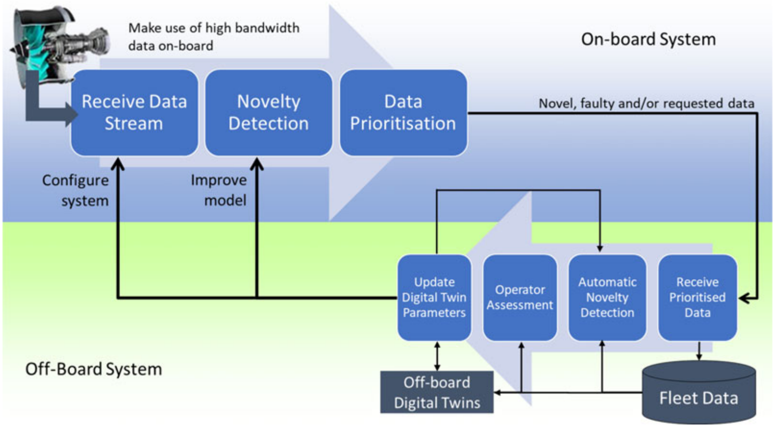

Figure 3. *Solution architecture. Data is continuously assessed for novelty by an on-board twin synchronised with a ground-based digital twin. Novel data is returned and triaged before re-training the on-ground twin.*

on-board computing system. The model is integrated into efficient input and output circular buffering functions that is accessed by the prioritisation decision logic.

Design assurance for the application was achieved via a combination of automated code analysis (via CLang format), automated testing, memory profiling (via Valgrind) and static analysis.

In order to evaluate performance, internal timers were used around the critical code sections in order to monitor data load and data evaluation times (model runs combined with data prioritisation), which are reported later in this paper.

2.2.6 Distributed digital twin updating

Figure 3 represents our closed-loop data acquisition architecture based on distributed digital twins. The asset specific digital twin parameters are synchronised with a ground-based counterpart so as to track evolving asset behaviour and maintain novelty detection performance. The increased off-board compute power is exploited to retrain the model with a training data set of historical asset and fleet data, which is augmented with new data once it is verified by fleet-based novelty assessment and operators [25]. New data are transferred to ground after an on-board assessment of high novelty or prediction uncertainty. For example, a new aircraft route may result in the engine operating (unfaulted) at new conditions where predicted behaviour must be extrapolated from the training data, and thus result in high prediction uncertainty. It is desirable to use this data to produce an updated set of model parameters through off-board training, and then use bidirectional communication to update the on-board model. The updated model will no longer need to extrapolate if similar operations are again performed, thus similar predictions will be confident and no data needs to be transferred to ground. The full set of off-board data (including that from other engines in the fleet) may be used, as is common practice, to perform fault diagnosis and monitoring of the received data. Additionally, the off-board digital twin may used for long-term asset trending and prediction of required maintenance events.

3.0 Data sets

The effectiveness data prioritisation is dependent on the performance of machine data novelty relative to its digital twin. Two different time-series data sets were used to validate the CNN model predictions for a variety of signal types and its performance in identifying data of high novelty.

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

Each data set was split into three subsets: training, validation and test. Test data was used solely for evaluation and, unless otherwise specified, consisted of the most recently recorded data. Training and validation data was drawn from earlier data randomly. This split setup mimics real-world applications where only past data is available and models must be deployed to an aircraft. In all cases, the training set was used for the determination of scaling, and the only data directly available to the networks for learning. The validation set was used for early stopping and for hyper-parameter tuning while the test set was kept solely for evaluation.

NASA turbofan degradation dataset [38]. This public dataset is made available from a prognostics competition aimed at assessing the quality of remaining useful life prediction algorithms. Specifically, subset “FD002” was used, which contains data on synthetically modelled gas turbine degradation over six different operating conditions. There are three input signals: altitude, air speed and thrust setting. The data simulate one measurement snapshot of each measured signal per flight to characterise an engine’s performance during that flight. For the selected dataset, 519 different engines with different initial health conditions were simulated over increasing synthetic degradation, with each engine termed a simulation ‘run’. Our training data contained only low degradation and no failures, while the test set had progressed to failure conditions near the end of the simulation run. Further, only the first half of each run was used during model training to only learn normal engine behaviours. We use this data to illustrate the ability of the model to represent an arbitrary time-series different from the high-bandwidth streamed data and detect the emergence of novelty.

In-service fault case study. This case study, provided by Rolls-Royce, includes continuous data 1 Hz from a twin-engine aircraft across 116 flights. These data, in contrast to the NASA dataset, contain engine dynamics and real-world disturbances, but with a relatively low level of degradation and only one engine. There is one known anomaly in this data (validated by domain experts to occur on the 113th flight) as well as several other potential fault symptoms that manifest as data anomalies discovered by the model. Only one engine from the aircraft is used for modelling so as to represent the real aircraft avionics, which do not allow real-time access to signals from both engines by design, though validation from both engines post-test was performed. The data presented here have been normalised and anonymised to protect the interests of parties that supplied it. In this paper, the dataset has been reduced to four signals for clarity, these being: engine fan speed (N1), compressor exit pressure (P30), high pressure turbine gas temperature (TGT) and compressor exit temperature (T30).

4.0 Results and discussion

4.1 NASA turbofan degradation

A model was trained for each of the 21 sensor signals contained in the dataset using the modelling approach described in Section 2.2. The three input signals as described in Section 3.0 were used as input to the models. The inputs cause the engine to operate in six distinct operating modes or conditions. Only data associated with nominal operation was used for training – and hence the digital twin represented the nominal response of the system. The data consist of one measurement snapshot per flight, the window length was therefore set to 1 i.e. only considering the most recent data and without information on engine dynamics.

Validation of the model accuracy was performed using data from the ‘low degradation’ validation dataset. The models achieved a mean absolute error (MAE) of 0.029 ± 0.034 (normalised range) across all 21 sensors and across all the engines. Testing was then performed on the remaining data.

In the test data, it is known that the turbofan engine degrades to a failure state at the end of each run. The growth in plant-model mismatch is measured by the novelty score $\tilde{d}$, given by Equation (12). This metric provides a measure of novelty resulting from the increase in degradation towards the end of each engine run. The novelty score $\tilde{d}$ is calculated over the outputs of each of the 21 models. Each engine run is separated into 10 equally sized segments of testing data (based on time). The change in the distribution in $\tilde{d}$ can clearly be seen to increase in terms of the mean, variance and maximum

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

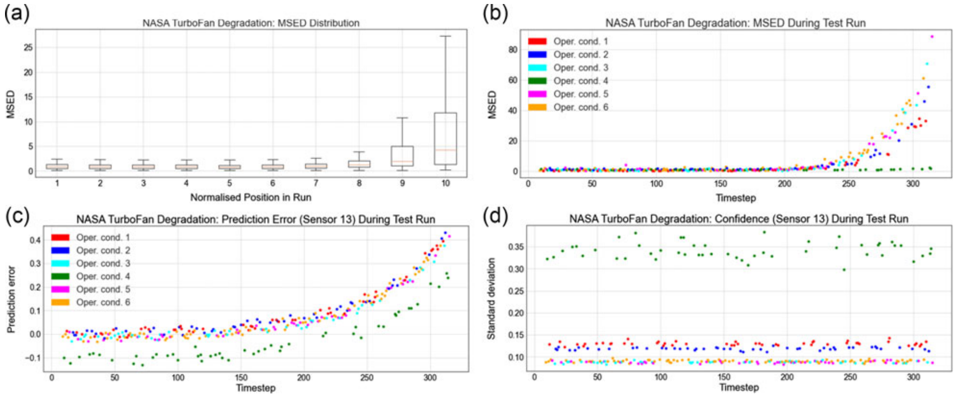

Figure 4. *(a) Distribution of the novelty score $\tilde{d}$ based on position within a test run. Larger means, variances and maximum values are observed towards the end of runs where faults occur. (b) Maximum values of the novelty score $\tilde{d}$ across all model outputs for a single engine run. The operating condition is indicated by colour. (c) Prediction error, y_n-μ_n, for Sensor 13 over a single engine run. (d) Standard deviation, σ_n, for model predictions on Sensor 13.*

values across each data segment, see Fig. 4a. In addition to a visual inspection, the distribution of $\tilde{d}$ is compared for the first and second half of each run contained in the test data, for each engine. The statistical comparison is made using a Mann-Whitney U test (since there is no guarantee of normality) under the null hypothesis that both sets came from the same distribution. The two sets of distributions were determined to have different shapes with a p value ≈ 0 ($< 10^{-308}$). The mean of the first part of the runs was 0.980 while the latter portion was 3.466, thus demonstrating that the model provides output allowing correct prioritisation of this data by predictive error. A large variance in $\tilde{d}$ is observed in the later data segments. This large variance is explained by an analysis of the operating conditions towards the end of the training data. It can be observed that model outputs associated with operating condition 4 are significantly lower than for the other operating conditions, as seen in Fig. 4b.

The comparatively low value of $\tilde{d}$ for operating condition 4 should not be confused with a good model prediction. In fact, the model predictions for data associated with operating condition 4 are consistently worse than for the other conditions, see Fig. 4c which shows the prediction error, $y_n - \mu_n$, for the model predictions on sensor 13 only. The standard deviation of the model predictions across operating condition 4 are significantly larger in comparison to the other conditions, Fig. 4d.

The novelty score for data under each operating condition, except for operating condition 4, increase as degradation increases. In operation of the digital twin-based novelty detection, this would correctly trigger an alert and identify the degradation. The models are bad predictors for operating condition 4 leading to a high uncertainty, which would cause these points to be prioritised for data capture in order to improve the model in the future. This is a desirable behaviour as it reduces false alerts where there is a lack of knowledge – even if in this case it misses degradation.

These results illustrate that the modelling process and prioritisation methodology are applicable to a time series of static snapshot generated by a time-varying process. The model flexibility allows the approach to be applied without structural changes to dynamic data, as shown in the next section. The above results can be recreated using the code that accompanies this manuscript.[1]

[1] The code to reproduce these models and results on the NASA data-set is made available alongside this paper: https://doi.org/10.15131/shef.data.21695048

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

Table 1. *Model prediction performance.*

Output Signal	MAE	Unit
N1	0.085	% of max speed
P30	0.715	psi
TGT	1.754	K
T30	0.872	K

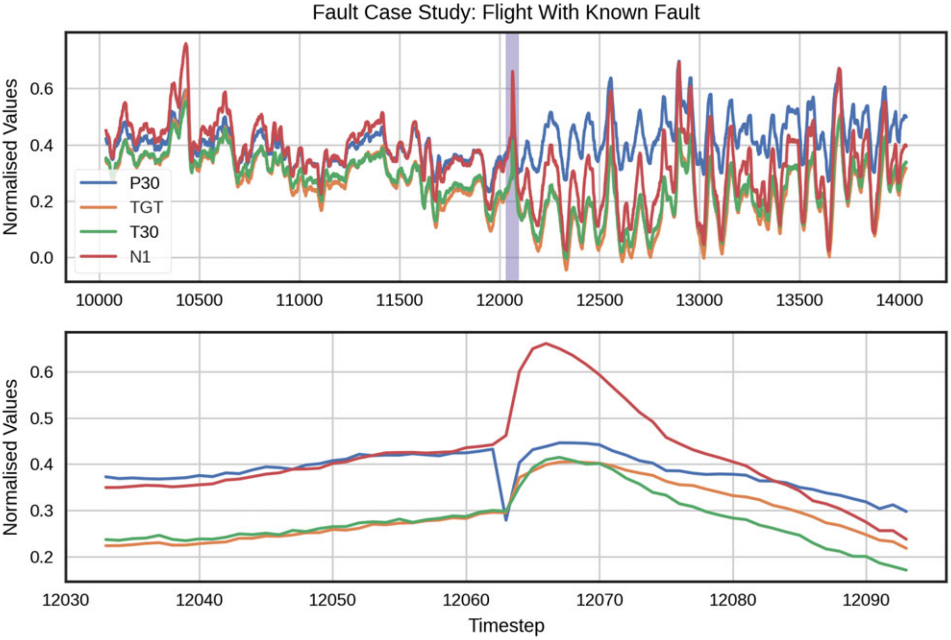

Figure 5. *Normalised signals for the flight with the confirmed fault. The highlighted timespan in the top graph is enlarged on the bottom graph. The spike in the internal pressure signal (P30) in the bottom graph is a result of a known fault as is all behaviour afterwards. The temperature T30 is an internal temperature detected in the same locale as P30, N1 is the low-speed shaft and TGT is the turbine gas temperature.*

4.2 In-service fault case study

This case study concerns the emergence of a real fault that was detected in service and has been confirmed by other health monitoring systems and domain experts at our industrial partner. No precursor warning of this fault was given by the existing health monitoring system. The symptoms were noted as being most prominent in the compressor exit pressure (P30) and related engine systems. The initial spike (at time point 12,000) is most prominent, although not unique in the dataset, and after the spike, engine behaviour changes to an anomalous mode for the rest of the flight, see Fig. 5. Four models were trained using the scheme described in Section 2.2.1, one for each of the four signals shown in Fig. 5. Given the fault was known to occur at 11th flight, the first 60 flights were used for training, thus avoiding fault data being included in the training set. The inputs to each model are the three signals not predicted for, see Appendix B for more details. The prediction performance was evaluated across the test set in terms of MAE in the original signal range. This demonstrates that the models performed exceptionally in terms of predictive error (see Table 1).

Several interesting properties of the solution are observed from the models' signal predictions at the time region around the known fault (see Fig. 6). Firstly, the fault, and the ensuing anomalous behaviour,

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

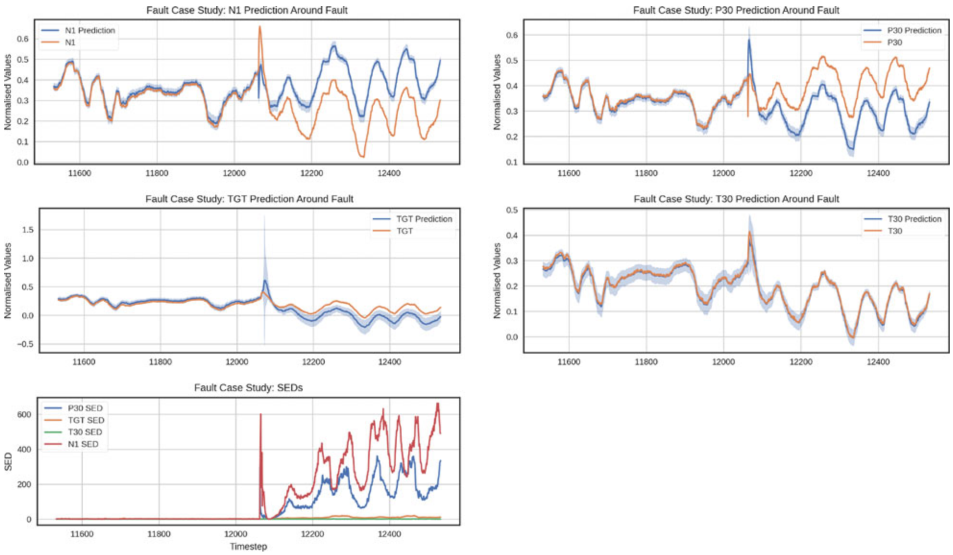

Figure 6. Top four graphs: prediction around the fault, in descending order of the novelty score $\tilde{d}$. Highlighted regions indicate a 95% confidence interval (predicted). Bottom graph: $\tilde{d}$ for each model around the fault.

is clearly detected by the model. Secondly, the dominant novelty score (highest magnitude) for this fault is observed in the model predicting N1, despite the fault being most prominent in the P30 signal. This indicates that it is possible for the model to detect anomalous behaviour in signals not being directly predicted. Further, this indicates that the solution could detect anomalies in a wide range of signals with a small number of models if the set of inputs and outputs is designed well.

In contrast, the T30 model (which also uses P30 as an input) is significantly more accurate in prediction, and therefore assesses the data as less anomalous. During the fault, T30 still maintains the same relationship with P30, so while using a signal as a model input may allow detection of faults that manifest in that signal, it does not guarantee it. The residual signature across all models is of high diagnostic value, in this example pressure and temperature (P30 and T30) are physically correlated, thus making a sensor fault the most likely fault cause.

Thirdly, the confidence intervals for all the models remain self-similar, with the exception of the P30 and TGT model during the initial spike, showing that the training data covers a sufficient range of operating conditions to allow signals to be accurately predicted in this region. The large confidence interval around the initial spike in these models demonstrates that this is the only section (from those tested) that includes data outside previously seen ranges. A large confidence interval may be an indication that something unusual is occurring; however, since these regions do not appear in training data, it is not reasonable for the system to identify them as highly anomalous unless the error is also very high.

The fault shown in Figs. 5 and 6 is the only fault in the data that has been previously detected; however, the solution was able to identify multiple potential precursor events that show the same signature as the known fault, but without the initial spike. These were detected in the data from three flights prior (Fig. 7), and six flights prior, to the known fault event occurring. These events were not detected by current health monitoring systems, however, compared to the other (assumed correctly functioning) engine on the aircraft, they present the same fault signature [39]. The high magnitude of $\tilde{d}$ for these precursor fault data segments ($\tilde{d} > 100$) means they are stored and ranked highly for transmission to ground for further analysis, and thus providing an opportunity to take early action to mitigate the emerging fault.

The novelty score for the flight containing the known fault (flight 113), as well as the following flights, are observed to be distributed differently from the normal flights, see Fig. 8. The precursor faults

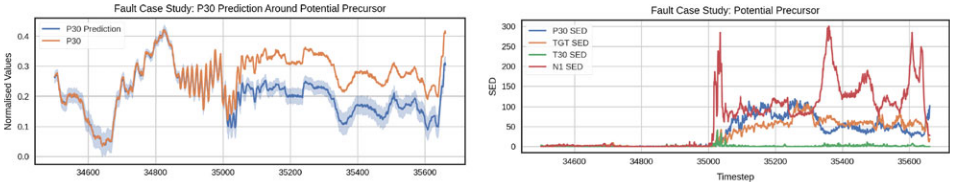

Figure 7. *Potential precursor fault, three flights before known fault.*

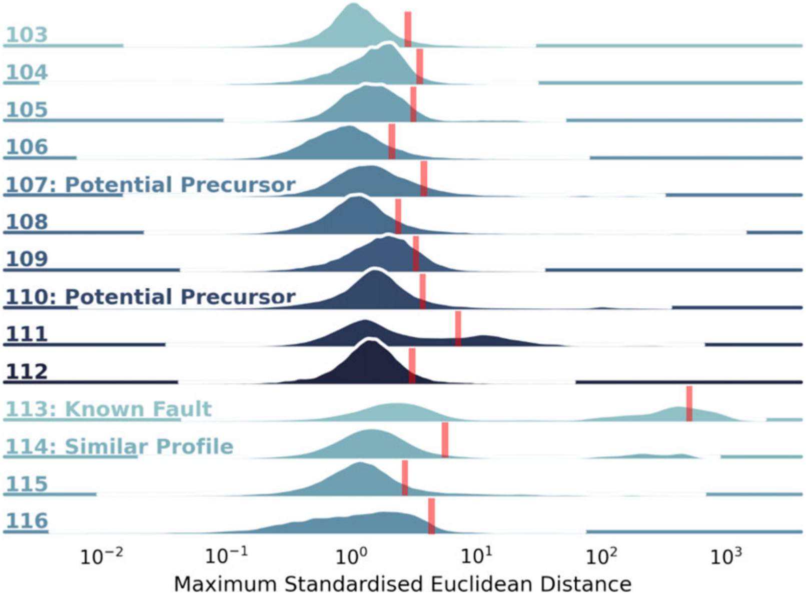

Figure 8. *Distribution of* $\tilde{d}$ *across test flights. The known fault has a large amount of mass at a much higher novelty scores than other flights, and potential precursors show smaller mass at similar novelty scores (*$\tilde{d} > 100$*). Flight 111 also exhibits high novelty score density due to irregularity in the shaft speed signal not found in other flights. Red lines represent a threshold for data collection, set by communications capacity. Anomalous data with high novelty score is prioritised.*

also show a similar, though less pronounced, shaped distribution with a second peak at $\tilde{d} > 100$. Data prioritisation is based on the novelty score, with a flight specific threshold (shown in red on the figure) for the data volume returned is set by the limited communication capacity. The data to the right of the threshold contains the unusual behaviours and is collected for off-board analysis.

Flight 111 also demonstrates an anomalous distribution; this is due to unusual behaviour in the shaft speed signal, which is not found in other flights. It is unclear whether this is a fault symptom or an unusual but harmless behaviour. This highlights the benefit of data prioritisation, over binary labelling since an operator can review the data to make the final determination on asset health. Under conventional on-board alerting schemes, a static threshold set to capture the faults and pre-cursors (at $\tilde{d} > 100$) would not alert to this unusual behaviour.

As well as identifying the potential faults, the system is also effective at collecting data from regions where training data is sparse (see Fig. 9). This figure shows the distribution of the training and test data of the in-service fault case study. Specifically, the distribution of the inputs of the network that predicts P30 are plotted in two dimensions after applying principal components analysis. In this plot the confidence interval increases in regions where the training data is sparse. In other words, the prediction uncertainty

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

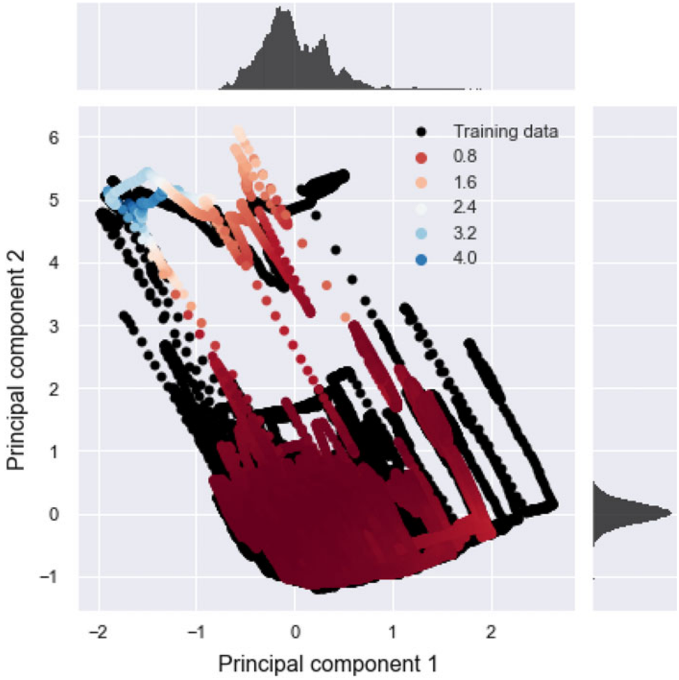

Figure 9. *P30 prediction confidence for different inputs. The test data is coloured based on confidence. Large values indicate a wide confidence interval. Black points show the distribution of the training data.*

increases for previously unseen inputs. Highly uncertain data are prioritised for collection and can be used to update the model to improve its prediction.

4.3 Real-time performance

The application was deployed on an industrial grade Xilinx Zynq XC7Z020, with a custom Linux OS running on the device’s ARM Cortex A7 microprocessor. In order to test real-time operation, the models designed above were deployed to the device and constrained to $<= 20\%$ of CPU time. The communication channel was assumed to be limited to only 50 data windows per ground data transmission, thus an output buffer of size 50 windows was configured. A synthetic data set of 25 signals over 100,000 time points was sent to the device.

The actual processing time for each run was recorded; a run consisted of all currently uploaded models being run once, the novelty score being determined and the output buffer being updated if necessary. The other primary resource-intensive operation, input retrieval and clean-up, was orders of magnitude smaller than that of a run ($\sim 25\mu s$), and not included in the timing calculation.

Figure 10 shows the distribution of run durations when the 20 models used to evaluate the NASA data set were used and Fig. 11 when the 4 models used to evaluate the in-service fault case study data set were used.

Both histograms have a long tail caused by the fact that the software runs on a non-real-time operating system (unpredictable system interrupts), which is why they are curtailed at the 95th quantile. These results show that the application is capable of operating in soft real-time e.g. all 20 of the NASA case study models could be run simultaneously on data at up to $\sim 50Hz$ and the fault case study models could be run at up to $\sim 500Hz$, including all overhead.

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

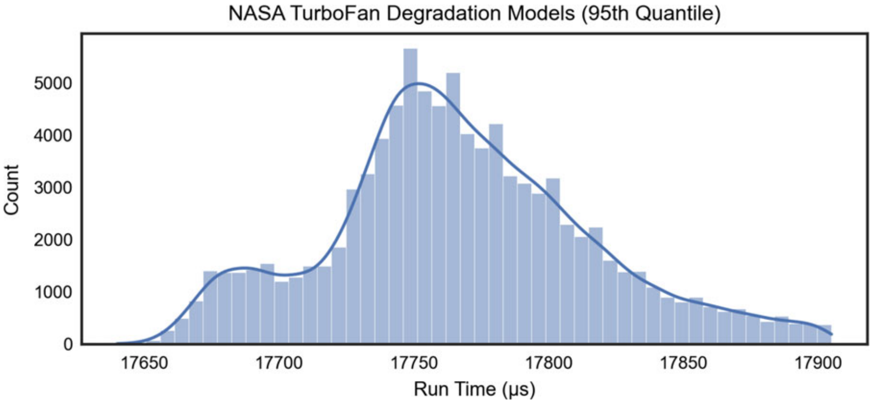

Figure 10. *95th quantile of run durations for the NASA turbofan degradation data. Smaller peak likely caused by cache hits.*

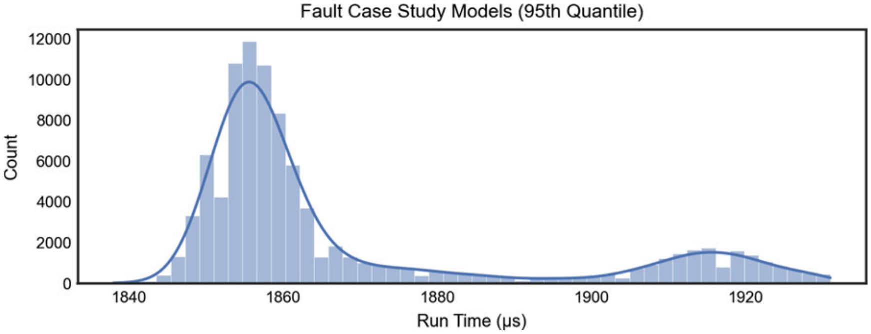

Figure 11. *95th quantile of run durations for the fault case study data. Smaller number of models and overall smaller model size relative to NASA data case study lead to the lower average run duration and likely the reversal of shape due to more cache hits.*

5.0 Conclusion

We have presented a data-driven data prioritisation design that runs in real-time on low-power embedded hardware. Prioritised data enables the updating of an on-ground digital twin and supports engine health monitoring decisions.

The system prioritises data based on real-signal prediction of an on-board digital twin constructed from an elastically sized ensemble of models. A flexible neural network design is presented that is used to build models for the ensemble from any input-output set and window length without the need for architecture changes. Execution on embedded system hardware has practically demonstrated a computationally inexpensive method for detection of unusual behaviour in real-time. The efficacy of the detection has been demonstrated on both real time-series and synthetic snapshot data, plus its real-time operation on embedded hardware has been proven and characterised. Further, the architecture is flexible and allows for deployment on new systems with minimal need for in-depth domain knowledge. The data prioritised by the on-board digital twin is returned to ground as part of a novel fleet monitoring architecture that overcomes communication bandwidth limiting the operator analysis of valuable health monitoring data.

Our research novelty is derived from its combination of a flexible modelling approach (allowing prediction of any input-output path within a machine), a confidence prediction for quantifying uncertainty in the models, and a custom CNN design that allows for real-time novelty analysis on embedded

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

hardware of data streams at 100s of samples per second over multiple signal channels. The solution will be deployed on-board in an upcoming flight test program, and future work will focus on analysis of the data returned by this test.

Funding. The Authors wish to acknowledge funding from Innovate UK under project E2EEHM (113095), and for the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

[1] Hofmann S., Duhovnikov S. and Schupke D. Massive data transfer from and to aircraft on ground: Feasibility and challenges. *IEEE Aerosp. Electron. Syst. Mag.*, 2021, **36**, (5), Conference Name: IEEE Aerospace and Electronic Systems Magazine, pp 6–14. ISSN: 1557-959X. doi: 10.1109/MAES.2021.3053119

[2] Li Y.G. and Nilkitsaranont P. Gas turbine performance prognostic for condition-based maintenance. *Appl. Energy*, 2009, **86**, (10), pp 2152–2161. ISSN: 0306-2619. doi: 10.1016/j.apenergy.2009.02.011. URL: http://www.sciencedirect.com/science/article/pii/S0306261909000506 (visted on 11/20/2020)

[3] King S., et al. *Equipment Health Monitoring in Complex Systems*. Artech House, London; 2017. ISBN: 1630814970, 9781630814977

[4] Glaessgen E. and Stargel D. The digital twin paradigm for future NASA and U.S. Air Force vehicles. In *53rd AIAA/ASME Structures, Structural Dynamics and Materials Conference*, American Institute of Aeroautics and Astronautics, 2012, p 118. doi: 10.2514/6.2012-1818. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2012-1818 (visited on 11/20/2020).

[5] Tao F., et al. Digital twin in industry: State-of-the-art. *IEEE Trans. Ind. Inf.*, 2018, **15**, (4), publisher: IEEE, pp 2405–2415.

[6] Asgari H., Chen X.Q. and Sainudiin R. Modelling and simulation of gas turbines. *Int. J. Modell., Identif. Control*, 2013, **20**, (3), publisher: Inderscience Publishers Ltd., pp 253–270.

[7] Panov V. Gasturbolib: Simulink library for gas turbine engine modelling. *Turbo Expo: Power Land, Sea, Air*, 2009, **48821**, pp 555–565.

[8] Willsky A.S. A survey of design methods for failure detection in dynamic systems. *Automatica*, 1976, **12**, pp 601–611. ISSN: 0005-1098. URL: http://ssg.mit.edu/~willsky/publ_pdfs/15_pub_AUTO.pdf

[9] Frank P.M. and Ding X. Frequency domain approach to optimally robust residual generation and evaluation for model-based fault diagnosis. *Automatica*, 1994, **30**, (5), pp 789–804. ISSN: 0005-1098. doi: 10.1016/0005-1098(94)90169-4. URL: https://www.sciencedirect.com/science/article/pii/0005109894901694 (visited on 02/05/2021).

[10] Frisk E. and Nielsen L. Robust residual generation for diagnosis including a reference model for residual behavior. *Automatica*, 2006, **42**, (3), pp 437–445. ISSN: 0005-1098. doi: 10.1016/j.automatica.2005.10.009. URL: https://www.sciencedirect.com/science/article/pii/S0005109805003948 (visited on 02/05/2021).

[11] Brotherton T., et al. *eSTORM: Enhanced Self Tuning On-Board Real-Time Engine Model*. Tech. rep. Section: Technical Reports. Intelligent Automation Corp Poway CA, 2003. URL: https://apps.dtic.mil/sti/citations/ADA511669 (visited on 02/05/2021).

[12] Kobayashi T. and Simon D.L. Hybrid Kalman filter approach for aircraft engine in-flight diagnostics: Sensor fault detection case. *Am. Soc. Mech. Eng. Digital Collect.*, 2008, pp 745–755. doi: 10.1115/GT2006-90870. URL: https://asmedigitalcollection.asme.org/GT/proceedings/GT2006/42371/745/314663 (visited on 02/05/2021).

[13] Zhao N., Wen X. and Li S. “A review on gas turbine anomaly detection for implementing health management”. *Turbo Expo: Power Land, Sea, Air*, 2016, **49682**. American Society of Mechanical Engineers, V001T22A009.

[14] M. Amozegar and Khorasani K. An ensemble of dynamic neural network identifiers for fault detection and isolation of gas turbine engines. *Neural Netwk.*, 2016, **76**, pp 106–121. ISSN: 0893-6080. doi: 10.1016/j.neunet.2016.01.003. URL: http://www.sciencedirect.com/science/article/pii/S0893608016000046 (visited on 11/26/2020).

[15] Zhong S., et al. A novel anomaly detection method for gas turbines using weight agnostic neural network search. In *2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM)*, IEEE, New York, USA, 2020, pp 1–6.

[16] Yan W. and Yu L. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. *Annu. Conf. PHM Soc.*, 2015, **7**, (1). doi: 10.36001/phmconf.2015.v7i1.2655

[17] Yan W. Detecting gas turbine combustor anomalies using semisupervised anomaly detection with deep representation learning. *Cognit. Comput.*, 2020, **12**, (2), tex.ids: yanDetectingGasTurbine2020a publisher: Springer, pp 398–411.

[18] Lee G., et al. Unsupervised anomaly detection of the gas turbine operation via convolutional auto-encoder. In *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, New York, USA, 2020, pp 1–6.

[19] Liu J., et al. Fault detection for gas turbine hot components based on a convolutional neural network. *Energies*, 2018, **11**, (8), publisher: Multidisciplinary Digital Publishing Institute, p 2149.

[20] Luo H. and Zhong S. Gas turbine engine gas path anomaly detection using deep learning with Gaussian distribution. In *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, IEEE, New York, USA, 2017, pp 1–6.

[21] Amontamavut P., Nakagawa Y. and Hayakawa E. Separated Linux process logging mechanism for embedded systems. In *2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, IEEE, New York, USA, 2012, pp 411–414.

[22] Summerville D.H., Zach K.M. and Chen Y. Ultra-lightweight deep packet anomaly detection for Internet of Things devices. In *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, IEEE, New York, USA, 2015, pp 1–8.

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

[23] Attia M.B., et al. On-device anomaly detection for resource-limited systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, New York, USA, 2015, pp 548–554.

[24] Nix D.A. and Weigend A.S. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, 1. June 1994, pp 55–60. doi: 10.1109/ICNN.1994.374138

[25] Montana F., et al. Through-life monitoring of resource-constrained systems and fleets. *arXiv preprint arXiv:2301.01017*, 2023.

[26] Rezk N.M., et al. Recurrent neural networks: An embedded computing perspective. IEEE Access, 2020, 8. arXiv: 1908.07062, pp 57967–57996. ISSN: 2169-3536. doi: 10.1109/ACCESS.2020.2982416. URL: http://arxiv.org/abs/1908.07062 (visited on 11/27/2020).

[27] Sze V., et al. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE*, 2017, **105**, (12), pp 2295–2329. doi: 10.1109/JPROC.2017.2761740

[28] Srivastava N., et al. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 2014, **15**, (1), publisher: JMLR. Org., pp 1929–1958.

[29] Iandola F.N., et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[30] Kingma D.P. and Ba J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Liu L., et al. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[32] Redmon J., et al. You only look once: Unified, real-time object detection. *arXiv:1506.02640 [cs]*, 2016. arXiv: 1506.02640 version: 5. URL: http://arxiv.org/abs/1506.02640 (visited on 05/12/2021).

[33] Gal Y. and Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *Int. Conf. Mach. Learn. PMLR*, 2016. URL: http://arxiv.org/abs/1506.02142 (visited on 05/12/2021).

[34] De Maesschalck R., Jouan-Rimbaud D. and Massart D.L. The Mahalanobis distance. *Chemometr. Intell. Lab. Syst.*, 2000, **50**, (1), pp 1–18. ISSN: 0169-7439. doi: 10.1016/S0169-7439(99)00047-7. URL: http://www.sciencedirect.com/science/article/pii/S0169743999000477 (visited on 01/04/2021).

[35] Abadi M., et al. "Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, Association for Computing Machinery, New York, 2016, pp 265–283. https://dl.acm.org/doi/10.5555/3026877.3026899

[36] Deng Y. Deep learning on mobile devices: A review. 2019, 10993, pp 52–66. ISSN: 1996756X. doi: 10.1117/12.2518469. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10993/109930A/Deep-learning-on-mobile-devices-a-review/10.1117/12.2518469.full

[37] Chen J. and Ran X. Deep learning with edge computing: A review. *Proc. IEEE*, 2019. ISSN: 15582256. doi: 10.1109/JPROC.2019.2921977

[38] Saxena A., et al. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, 2008, pp 1–9. doi: 10.1109/PHM.2008.4711414

[39] Jacobs W., et al. Inter-engine variation analysis for health monitoring of aerospace gas turbine engines. *PHM Soc. Eur. Conf.*, 2018, **4**, (1), p 1. ISSN: 2325-016X. doi: 10.36001/phme.2018.v4i1.444. URL: https://papers.phmsociety.org/index.php/phme/article/view/444 (visited on 05/17/2021).

Appendix A Network reproduction

The loss function (Equation 11) was simplified and reordered for batch-wise implementation since the model's confidence output $\alpha = log(\sigma)$ rather than σ:

$$\mathcal{LL} = \frac{1}{2}\sum_{n=1}^{N}\left(\frac{y_n - \mu}{e^{\alpha_n}}\right)^2 - \sum_{n=1}^{N}\alpha_n - \frac{1}{2}log(2\pi) \tag{13}$$

where N is the batch size. The final term is constant and could therefore be dropped for computational efficiency, although it was kept during our work to keep the meaning of the equation intact.

Important network hyper-parameters not mentioned elsewhere:

- LReLU $\alpha = 0.3$
- Dropout rate $= 0.5$
- Convolutional layers
    - Each used 64 filters in the NASA study
    - Each used 32 filters in the fault case study
        - An optimisation found to cause minimal performance loss while reducing computation
    - Padding irrelevant for 1x1 blocks, 'same' used internally
    - No padding for temporal and spatial fold blocks

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

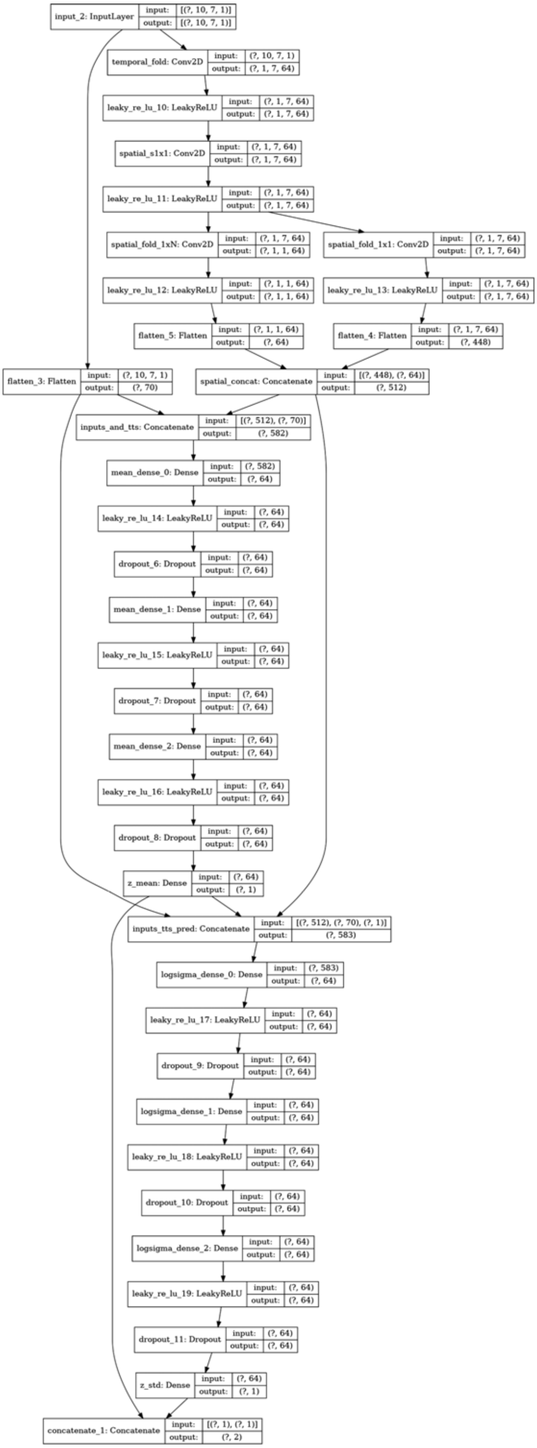
input_2: InputLayer | input: [(?, 10, 7, 1)] | output: [(?, 10, 7, 1)]
temporal_fold: Conv2D | input: (?, 10, 7, 1) | output: (?, 1, 7, 64)
leaky_re_lu_10: LeakyReLU | input: (?, 1, 7, 64) | output: (?, 1, 7, 64)
spatial_s1x1: Conv2D | input: (?, 1, 7, 64) | output: (?, 1, 7, 64)
leaky_re_lu_11: LeakyReLU | input: (?, 1, 7, 64) | output: (?, 1, 7, 64)
spatial_fold_1xN: Conv2D | input: (?, 1, 7, 64) | output: (?, 1, 1, 64)
spatial_fold_1x1: Conv2D | input: (?, 1, 7, 64) | output: (?, 1, 7, 64)
leaky_re_lu_12: LeakyReLU | input: (?, 1, 1, 64) | output: (?, 1, 1, 64)
leaky_re_lu_13: LeakyReLU | input: (?, 1, 7, 64) | output: (?, 1, 7, 64)
flatten_5: Flatten | input: (?, 1, 1, 64) | output: (?, 64)
flatten_4: Flatten | input: (?, 1, 7, 64) | output: (?, 448)
flatten_3: Flatten | input: (?, 10, 7, 1) | output: (?, 70)
spatial_concat: Concatenate | input: [(?, 448), (?, 64)] | output: (?, 512)
inputs_and_tts: Concatenate | input: [(?, 512), (?, 70)] | output: (?, 582)
mean_dense_0: Dense | input: (?, 582) | output: (?, 64)
leaky_re_lu_14: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_6: Dropout | input: (?, 64) | output: (?, 64)
mean_dense_1: Dense | input: (?, 64) | output: (?, 64)
leaky_re_lu_15: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_7: Dropout | input: (?, 64) | output: (?, 64)
mean_dense_2: Dense | input: (?, 64) | output: (?, 64)
leaky_re_lu_16: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_8: Dropout | input: (?, 64) | output: (?, 64)
z_mean: Dense | input: (?, 64) | output: (?, 1)
inputs_tts_pred: Concatenate | input: [(?, 512), (?, 70), (?, 1)] | output: (?, 583)
logsigma_dense_0: Dense | input: (?, 583) | output: (?, 64)
leaky_re_lu_17: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_9: Dropout | input: (?, 64) | output: (?, 64)
logsigma_dense_1: Dense | input: (?, 64) | output: (?, 64)
leaky_re_lu_18: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_10: Dropout | input: (?, 64) | output: (?, 64)
logsigma_dense_2: Dense | input: (?, 64) | output: (?, 64)
leaky_re_lu_19: LeakyReLU | input: (?, 64) | output: (?, 64)
dropout_11: Dropout | input: (?, 64) | output: (?, 64)
z_std: Dense | input: (?, 64) | output: (?, 1)
concatenate_1: Concatenate | input: [(?, 1), (?, 1)] | output: (?, 2)

Figure 12. *Neural network architecture with seven input signals and a 10 second window.*

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press

        - ∗ Used to reduce the size of the input
        - ∗ Equivalent to 'valid' padding
- Each dense layer had 64 hidden units for the signals tested
- Early stopping patience was 10 epochs without improvement
- RAdam parameters (adjusted based on validation data for each signal)
    - Learning rate $1e-3$ to $1e-5$
    - Min learning rate $1e-5$ to $1e-7$
    - Total steps $50,000$ to $200,000$
    - Warm-up proportion 0.1 to 0.5

An example implementation of the architecture used on our testing for seven inputs, looking at 10 seconds of data per window is shown in Fig. 12.

Appendix B Notes on data preprocessing

The same process was applied to each dataset:

- Data was scaled to the range [0,1]
    - Based on 2% to 98% quantiles in the training set

Appendix C In-service fault case study

The four models were:

- P30 (internal pressure) from outside temperature, outside pressure and N1
- TGT (Turbine Gas Temperature) from outside temperature, outside pressure, P30 and N1
- T30 (internal temperature) from outside temperature, outside pressure, P30 and N1
- N1 (speed of low speed shaft) from outside temperature, outside pressure and P30

Cite this article: Hartwell A., Montana F., Jacobs W., Kadirkamanathan V., Ameri N. and Mills A. R. Distributed digital twins for health monitoring: resource constrained aero-engine fleet management. *The Aeronautical Journal*, https://doi.org/10.1017/aer.2024.23

https://doi.org/10.1017/aer.2024.23 Published online by Cambridge University Press