Robustness Verification of Deep Neural Networks using Star-Based Reachability Analysis with Variable-Length Time Series Input

Neelanjana Pal[1][0000−0002−5978−8168], Diego Manzanas Lopez[1][0000−0003−0721−1241], and Taylor T Johnson[1][0000−0001−8021−9923]

Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37212, USA {neelanjana.pal,diego.manzanas.lopez,taylor.johnson}@vanderbilt.edu

Abstract. Data-driven, neural network (NN) based anomaly detection and predictive maintenance are emerging as important research areas. NN-based analytics of time-series data provide valuable insights and statistical evidence for diagnosing past behaviors and predicting critical parameters like equipment's remaining useful life (RUL), state-of-charge (SOC) of batteries, etc. Unfortunately, input time series data can be exposed to intentional or unintentional noise when passing through sensors, making robust validation and verification of these NNs a crucial task. Using set-based reachability analysis, this paper presents a case study of the formal robustness verification approach for time series regression NNs (TSRegNN). It utilizes variable-length input data to streamline input manipulation and enhance network architecture generalizability. The method is applied to two data sets in the Prognostics and Health Management (PHM) application areas: (1) SOC estimation of a Lithium-ion battery and (2) RUL estimation of a turbine engine. Finally, the paper introduces several performance measures to evaluate the effect of bounded perturbations in the input on network outputs, i.e., future outcomes. Overall, the paper offers a comprehensive case study for validating and verifying NN-based analytics of time-series data in real-world applications, emphasizing the importance of robustness testing for accurate and reliable predictions, especially considering the impact of noise on future outcomes.

Keywords: Predictive Maintenance · Time Series Data · Neural Network Verification · Star-set · Reachability Analysis · Noise · Robustness Verification · Prognostics and Health Management ·

1 Introduction

Over time, Deep Neural Networks (DNNs) have shown tremendous potential in solving complex tasks, such as image classification, object detection, speech recognition, natural language processing, document analysis, etc., sometimes even outperforming humans [15–17]. This has motivated a spurt in investigating the applicability of DNNs in numerous real-world applications, such as biometrics authentication, face authentication for mobile locking systems, malware detection, etc. In dealing with such susceptible information in these critical areas,

safety, security, and verification thereof have become essential design considerations.

Unfortunately, it has been demonstrated that minimal perturbations in the input can easily deceive state-of-the-art well-trained networks, leading to erroneous predictions [11,23,36]. The most researched domain for verification of such networks involves image inputs, particularly safety and robustness checking of various classification neural networks [4,7,12,22,38,41]. Previous research has analyzed feed-forward neural networks (FFNN [39]), convolutional neural networks (CNN [38]), and semantic segmentation networks (SSN [41]) using different set-based reachability tools, such as Neural Network Verification (NNV [19,42]) and JuliaReach [5], among others.

However, input perturbations are not only confined to image-based networks but also have been extended to other input types, including time series data or input signals with different noises in predictive maintenance applications [8, 43]. One such use case is in the manufacturing industry, where data from process systems, such as IoT sensors and industrial machines, are stored for future analysis [10,31]. Data analytics in this context provide insights and statistical information and can be used to diagnose past behavior [20, 46], and predicts future behavior [6,18,35], maximizing industry production. This application is not only limited to manufacturing, but is also relevant in fields like healthcare digitalization [37, 45] and smart cities [33, 34]. Noisy input data, here, refers to data containing errors, uncertainties, or disturbances, caused by factors like sensor measurement errors, environmental variations, or other noise sources.

While NN applications with image data have received significant attention, little work has been done in the domain of regression-type model verification, particularly with time series data in predictive maintenance applications. Regression-based models with noisy data are crucial for learning data representations and predicting future values, enabling fault prediction and anomaly detection in high-confidence, safety-critical systems [13, 29]. This motivated us to use verification techniques to validate the output of regression networks and ensure that the output(s) fall within a specific safe and acceptable range.

Contributions.

1. In this paper, we primarily focus on exploring a new case study, specifically examining time-series-based neural networks in two distinct industrial predictive maintenance application domains. We utilize the established concept of star-set-based reachability methods to analyze whether the upper and lower bounds of the output set adhere to industrial guidelines' permissible bounds. We develop our work[1] as an extension of the NNV tool to formally analyze and explore regression-based NN verification for time series data using sound and deterministic reachability methods and experiment on different discrete time signals to check if the output lies within pre-defined safe bounds.

[1] The code is available at: https://github.com/verivital/nnv/tree/master/code/nnv/examples/Submission/FMICS2023

2. Another significant contribution of our work is the flexibility of variable-length inputs in neural networks. This approach simplifies input manipulation and enhances the generalizability of network architectures. Unlike published literature that relied on fixed-sized windows [9, 24], which necessitated preprocessing and experimenting with window sizes, our method allows for flexibility in utilizing any sequence length. This flexibility improves the generalizability of reachability analysis.
3. We run an extensive evaluation on two different network architectures in two different predictive maintenance use cases. In this paper, we have introduced a novel robustness measure called Percentage Overlap Robustness (POR). Unlike the existing Percentage Sample Robustness (PR/PSR) [41], which considers only instances where reachable bounds remain entirely within permissible bounds, the proposed POR accounts for all instances with any possible overlap.
4. Finally, we develop insights on evaluating the reachability analysis on those networks and possible future direction.

Outline. The paper is organized as follows: Section 2 provides the necessary context for the background; Section 3 details the adversarial noises considered; Section 4 defines the verification properties; Section 6 explains the reachability calculations for layers to accommodate variable-length input; Section 5 defines the research problem, and Section 7 describes the methodology, including dataset, network models, and input attacks. Section 8 presents the experimental results, evaluation metrics, and their implications. Finally, Section 9 summarizes the main findings and suggests future research directions.

2 Preliminaries

This section introduces some basic definitions and descriptions necessary to understand the progression of this paper and the necessary evaluations on time series data.

2.1 Neural Network Verification Tool and Star Sets

The Neural Network Verification (NNV) tool is a framework for verifying the safety and robustness of neural networks [19, 42]. It analyzes neural network behavior under various input conditions, ensuring safe and correct operation in all cases. NNV supports reachability algorithms like the over-approximate star set approach [38, 40], calculating reachable sets for each network layer. These sets represent all possible network states for a given input, enabling the verification of specific safety properties. NNV is particularly valuable for safety-critical applications, such as autonomous vehicles and medical devices, ensuring neural networks are trustworthy and reliable under all conditions, and maintaining public confidence. For this paper, we have implemented the work as an extension of NNV tool and used the star [Def. 1] based reachability analysis to get the reachable sets of the neural networks at the outputs.

Definition 1. A generalized star set *(or simply star)* Θ *is a tuple* $\langle c, V, P\rangle$ *where* $c \in \mathbb{R}^n$ *is the center,* $V = \{v_1, v_2, \cdots, v_m\}$ *is a set of* m *vectors in* $\mathbb{R}^n$ *called basis vectors, and* $P : \mathbb{R}^m \to \{\top, \bot\}$ *is a predicate. The basis vectors are arranged to form the star's* $n \times m$ *basis matrix. The set of states represented by the star is given as:*

$$[\![\Theta]\!] = \{x \mid x = c + \Sigma_{i=1}^{m}(\alpha_i v_i) \text{ and } P(\alpha_1, \cdots, \alpha_m) = \top\}. \tag{1}$$

In this work, we restrict the predicates to be a conjunction of linear constraints, $P(\alpha) \triangleq C\alpha \leq d$ *where, for* p *linear constraints,* $C \in \mathbb{R}^{p\times m}$, α *is the vector of* m*-variables, i.e.,* $\alpha = [\alpha_1, \cdots, \alpha_m]^T$, *and* $d \in \mathbb{R}^{p\times 1}$.

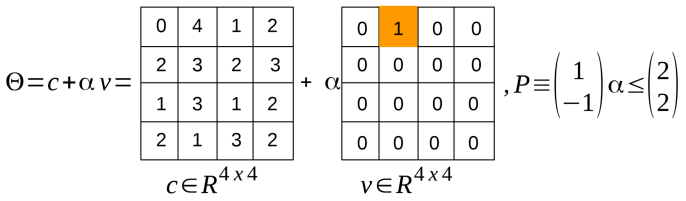

Fig. 1: Star for Time-series Data with four Feature Values (rows) with four time-steps (columns)

A 4×4 time series data with a bounded disturbance $b \in [-2, 2]$ applied on the time instance 2 of feature 1, i.e., position (1, 2) can be described as a Star depicted in Fig. 1.

2.2 Time Series and Regression Neural Network

Signal. The definition of a 'signal' varies depending on the applicable fields. In the area of signal processing, a ***signal*** S can be defined as some physical quantity that varies with respect to (w.r.t.) some independent dimension (e.g., space or time) [27]. In other words, a signal can also be thought of as a function that carries information about the behavior of a system or properties of some physical process [28].

$$S = g(q) \tag{2}$$

where q is space, time, etc. Depending on the nature of the spaces signals are defined over, they can be categorized as discrete or continuous. Discrete-time signals are also known as time series data.
We next define the specific class of signals considered in this paper, namely time series.

Definition 2. *A* ***time series signal*** S_T *is defined as an ordered sequence of values of a variable (or variables) at different time steps. In other words, a*

time series signal is an ordered sequence of discrete-time data of one or multiple features[2].

$$\begin{aligned} S_T &= s_{t_1}, s_{t_2}, s_{t_3}, ... \\ T &= t_1, t_2, t_3, ... \end{aligned} \tag{3}$$

where, $t_1, t_2, t_3, \ldots$ *is an ordered sequence of instances in time* T *and* $S_T = s_{t_1}, s_{t_2}, s_{t_3}, \ldots$ *are the signal values at those time instances for each* $t = t_i$.

Here, sometimes we have used 'signal' to refer to the 'time series signal.'

Next, we define the specific types of neural networks considered in this paper, namely regression neural networks (specifically time series regression neural networks).

Definition 3. *A* ***time series regression neural network (TSRegNN)*** f *is a nonlinear/partially-linear function that maps each time-stamped value* $x(i, j)$ *(for* i^{th} *feature and* j^{th} *timestamp) of a single or multifeatured time series input* $\boldsymbol{x}$ *to the output* $\boldsymbol{y}$.

$$f: \ \boldsymbol{x} \in \mathbb{R}^{n_f \times t_s} \rightarrow \boldsymbol{y} \in \mathbb{R}^{p \times q} \tag{4}$$

where t_s, n_f *are the time-sequence length and the number of features of the input data, respectively,* $(j, i) \in \{1, \ldots, t_s\} \times \{1, \ldots, n_f\}$ *are the time steps and corresponding feature indices, respectively, and* p *is the number of values present in the output, while* q *is the length of each of the output values; it can either be equal to* t_s *or not, depending on the network design.*

Here, each row of $\boldsymbol{x}$ *represents a timestamped feature variable.*

2.3 Reachability of a Time Series Regression Network

In this section, we provide a description of how the reachability of a NN layer and the NN as a whole is computed for this study.

Here, we employ an alternative approach to define a Star set for time series data. It involves using the upper and lower bounds of the noisy input, centering the actual input. These bounds on each input parameter, along with the predicates, create the complete set of constraints the optimizer will solve to generate the initial set of states.

Definition 4. *A* ***layer*** L *of a TSRegNN is a function* $h: \ u \in \mathbb{R}^j \rightarrow v \in \mathbb{R}^p$, *with input* $u \in R^j$ *and output* $v \in R^p$ *defined as follows*

$$v = h(u) \tag{5}$$

where the function h *is determined by parameters* θ, *typically defined as a tuple* $\theta = \langle \sigma, W, b \rangle$ *for fully-connected layers, where* $W \in R^{j \times p}, b \in R^p$, *and activation function* $\sigma : R^j \rightarrow R^p$. *Thus, the fully connected NN layer is described as*

$$v = h(u) = \sigma(\boldsymbol{W} \times u + \boldsymbol{b}) \tag{6}$$

For convolutional NN-Layers, θ *may include parameters like the filter size, padding, or dilation factor, and the function in Eq. 6 may need alterations.*

[2] Each **feature** is a measurable piece of data that is used for analysis.

Definition 5. *Let $h: \ u \in \mathbb{R}^j \rightarrow v \in \mathbb{R}^p$, be a NN layer as described in Eq. 5. The **reachable set** R_h, with input, $I \in R^n$ is defined as*

$$\mathcal{R}_h \triangleq \{v \mid v = h(u), \ u \in \mathcal{I}\} \tag{7}$$

Reachability analysis (or shortly, reach) of a TSRegNN f on Star input set I is similar to the reachable set calculations for CNN [38] or FFNN [39], the only difference being both the previous works had been done for classification networks.

$$Reach(f, I): \ I \rightarrow \mathcal{R}_{ts} \tag{8}$$

We call $\mathcal{R}_{ts}(I)$ the *output reachable set* of the TSRegNN corresponding to the input set I.

For a regression type NN, the output reachable set can be calculated as a step-by-step process of constructing the reachable sets for each network layer.

$$\begin{aligned}
\mathcal{R}_{L_1} &\triangleq \{v_1 \mid v_1 = h_1(x), \ x \in \mathcal{I}\}, \\
\mathcal{R}_{L_2} &\triangleq \{v_2 \mid v_2 = h_2(v_1), \ v_1 \in \mathcal{R}_{L_1}\}, \\
&\vdots \\
\mathcal{R}_{ts} = \mathcal{R}_{L_k} &\triangleq \{v_k \mid v_k = h_k(v_{k-1}), \ v_{k-1} \in \mathcal{R}_{L_{k-1}}\},
\end{aligned}$$

where h_k is the function represented by the k^{th} layer L_k. The reachable set $\mathcal{R}_{L_k}$ contains all outputs of the neural network corresponding to all input vectors x in the input set $\mathcal{I}$.

3 Adversarial Noise

In the case of time series samples, while the sensor transmits the sampled data, sensor noises might get added to the original data. One example of such noise is sensor vibration, but sometimes the actual sources are not even known by the sensor providers [21].

Definition 6. *A **noise** can be defined as some unintentional, usually small-scaled signal which, when added to the primary signal, can cause malfunctioning of the equipment in an industrial premise. Mathematically, a noisy signal s^{noise} can be produced by a linear parameterized function $g_{\epsilon, s^{noise}}(\cdot)$ that takes an input signal and produces the corresponding noisy signal.*

$$s^{noise} = g_{\epsilon, s^{noise}}(s) = s + \Sigma_{i=1}^{n} \epsilon_i \cdot s_i^{noise} \tag{9}$$

For time series data, we can also assume the noise as a set of unit vectors associated with a coefficient vector ϵ at each time step i, where the value of the coefficient vector ϵ is unknown but bounded within a range $[\underline{\epsilon}, \overline{\epsilon}]$, i.e., $\underline{\epsilon_i} \leq \epsilon_i \leq \overline{\epsilon_i}$.

Types of Possible Noises. For an input sequence with t_s number of time instances and n_f number of features, there can be four types of noises (l_∞ norm) [A.1], based on its spread on the signal. They can be categorized as below:

1. **Single Feature Single-instance Noise (SFSI)** i.e., perturbing a feature value only at a particular instance (t) by a certain percentage around the actual value.
$$s^{noise} = g_{\epsilon, s^{noise}}(s) = s + \epsilon_t \cdot s_t^{noise} \tag{10}$$
2. **Single Feature All-instances Noise (SFAI)** i.e., perturbing a specific feature throughout all the time instances by a certain percentage around the actual values of a particular feature.
$$s^{noise} = g_{\epsilon, s^{noise}}(s) = s + \Sigma_{i=1}^{n} \epsilon_i \cdot s_i^{noise} \tag{11}$$
3. **Multifeature Single-instance Noise (MFSI)** i.e., perturbing all feature values but only at a particular instance (t), following Eq. 10 for all features.
4. **Multifeature All-instance Noise (MFAI)** i.e., perturbing all feature values throughout all the instances, following Eq. 11 for all features.

A sample plot for all four types of noises is shown in [A.8].

4 Verification Properties

Verification properties can be categorized into two types: local properties and global properties. A local property is defined for a specific input x at time-instance t or a set of points X in the input space $R^{n_f \times t_s}$. In other words, a local property must hold for certain specific inputs. On the other hand, a global property [44] is defined over the entire input space $R^{n_f \times t_s}$ of the network model and must hold for all inputs without any exceptions.

Robustness. Robustness refers to the ability of a system or a model to maintain its performance and functionality under various challenging conditions, uncertainties, or perturbations. It is a desirable quality that ensures the system's reliability, resilience, and adaptability in the face of changing or adverse circumstances. For an input perturbation measured by δ and admissible output deviation ϵ, the 'delta-epsilon' formulation for the desired robustness property can be written as:

$$||x' - x||_\infty < \delta \implies ||f(x') - f(x)||_\infty < \epsilon \tag{12}$$

where x is the original input belonging to the input space $R^{n_f \times t_s}$, x' is the noisy input, $f(x')$ and $f(x)$ are NN model outputs for, respectively, x' and x, δ is the max measure of the noise added, ϵ is the max deviation in the output because of the presence of noise ($\delta, \epsilon \in \mathbf{R} > 0$).

Local Robustness. Given a TSRegNN f and an input time series signal S, the network is called **locally robust** to any noise $\mathcal{A}$ if and only if: the estimated output reachable bounds for a particular time-step corresponding to the noisy input lie between predefined allowable bounds w.r.t to the actual signal.

Robustness Value (RV) of a time series signal S is a binary variable, which indicates the local robustness of the system. RV is 1 when the estimated output range for a particular time instance (t) lies within the allowable range, making it locally robust at t; otherwise, RV is 0.

$RV = 1 \iff {LB_t}^{est} \geq {LB_t}^{allow} \wedge {UB_t}^{est} \leq {UB_t}^{allow}$ else, RV = 0

where ${LB_t}^{est}$ and ${UB_t}^{est}$ are estimated bounds and ${LB_t}^{allow}$ and ${UB_t}^{allow}$ are allowable bounds.

Definition 7. ***Percentage Sample Robustness (PR)*** *of a TSRegNN corresponding to any noisy input is defined as*

$$PR = \frac{N_{robust}}{N_{total}} \times 100\%, \tag{13}$$

where N_{robust} is the total number of robust time instances, and N_{total} = the total number of time steps in the time series signal. Percentage robustness can be used as a measure of ***global robustness*** *[44] of a TSRegNN w.r.t any noise.*

In this study, we adapt the concept of Percentage Robustness (PR) previously used in image-based classification or segmentation neural networks [41] to time-series inputs. PR in those cases assessed the network's ability to correctly classify/segment inputs even with input perturbations for a given number of images/pixels. We extend this concept to analyze the robustness of time-series inputs in our research.

Definition 8. ***Percentage Overlap Robustness (POR)*** *of a TSRegNN corresponding to any noisy input is defined as*

$$POR = \frac{\Sigma_{i=1}^{N_{total}}(PO_i)}{N_{total}} \times 100\%, \tag{14}$$

where N_{total} = total number of time instances in the time series signal, and PO_i is the percentage overlap between estimated and allowed ranges at each time step w.r.t the estimated range

$$PO = \frac{Overlapped\ Range}{Estimated\ Range} \tag{15}$$

Here Overlapped Range is the overlap between the estimated range and the allowable range for a particular time step. The Allowable Range indicates the allowable upper and lower bounds, whereas Estimated Range is the output reachable bounds given by the TSRegNN for that time step. Percentage overlap robustness can also be used as a measure of ***global robustness*** *[44] of TSRegNN.*

When selecting robustness properties, it is crucial to consider the specific application area. If the application allows for some flexibility in terms of performance, POR can be utilized. On the other hand, if the application requires a more conservative approach, PR should be considered. An example showing calculations for the robustness measures is shown in [A.1].

Monotonicity. In PHM applications, the monotonicity property refers to the system's health indicator, i.e., the degradation parameter exhibiting a consistent increase or decrease as the system approaches failure. PHM involves monitoring a system's health condition and predicting its Remaining Useful Life (RUL) to enable informed maintenance decisions and prevent unforeseen failures. For detailed mathematical modeling of the monotonicity property, please refer to [32] and the latest report on formal methods at [9]. In general, for a TSRegNN $f: \mathbf{x} \in \mathbb{R} \rightarrow \mathbf{y} \in \mathbb{R}$ with single-featured input and output spaces, at any time instance t, the property for monotonically decreasing output can be written as:

$$\begin{aligned} \forall x' \exists \delta : x \leq x' \leq x + \delta \implies f(x') \leq f(x) \\ \forall x' \exists \delta : x - \delta \leq x' \leq x \implies f(x') \geq f(x) \end{aligned} \tag{16}$$

This is a local monotonicity property. If this holds true for the entire time range, then the property can be considered as a global property [44]. In this paper, the monotonicity property is only valid for the PHM examples for RUL estimation.

5 Robustness Verification Problem Formulation

We consider the verification of the robustness and the monotonicity properties.

*Problem 1 (**Local Robustness Property**).* Given a TSRegNN f, a time series signal S, and a noise $\mathcal{A}$, prove if the network is locally robust or non-robust [Sec. 4] w.r.t the noise $\mathcal{A}$; i.e., if the estimated bounds obtained through the reachability calculations lie within the allowable range of the actual output for the particular time instance.

*Problem 2 (**Global Robustness Property**).* Given a TSRegNN f, a set of N consecutive time-series signal $\mathbf{S} = \{S_1, \ldots, S_N\}$, and a noise $\mathcal{A}$, compute the percentage robustness values (PR [Def. 7] and POR [Def. 8]) corresponding to $\mathcal{A}$.

*Problem 3 (**Local Monotonicity Property**).* Given a TSRegNN f, a set of N consecutive time-series signal $\mathbf{S} = \{S_1, \ldots, S_N\}$, and a noise $\mathcal{A}$, show that both the estimated RUL bounds of the network [Eq. 16] corresponding to noisy input S'_t at any time instance t are monotonically decreasing.

To get an idea of the global performance [44] of the network, local stability properties have been formulated and verified for each point in the test dataset for 100 consecutive time steps.

The core step in solving these problems is to solve the local properties of a TSRegNN f w.r.t a noise $\mathcal{A}$. It can be done using over-approximate reachability analysis, computing the 'output reachable set' $\mathcal{R}_t s = Reach(f, I)$ that provides an upper and lower bound estimation corresponding to the noisy input set I.

In this paper, we propose using percentage values as robustness measures for verifying neural networks (NN). We conduct reachability analysis on the output set to ensure it stays within predefined safe bounds specified by permissible

upper-lower bounds. The calculated overlap or sample robustness, expressed as a percentage value, represents the NN's robustness achieved through the verification process under different noise conditions. The proposed solution takes a sound and incomplete approach to verify the robustness of regression neural networks with time series data. The approach over-approximates the reachable set, ensuring that any input point within the set will always have an output point contained within the reachable output set (sound [Def. 9]). However, due to the complexities of neural networks and the over-approximation nature of the approach, certain output points within the reachable output set may not directly correspond to specific input points (incomplete [Def. 10]). Over-approximation is commonly used in safety verification and robustness analysis of complex systems due to its computational efficiency and reduced time requirements compared to exact methods.

6 Reachability of Specific Layers to Allow Variable-Length Time Series Input

Reachability Of A Fully-connected Layer. We consider a fully-connected layer with the following parameters: the weights $W_{fc} \in R^{op \times ip}$ and the bias $b_{fc} \in R^{op \times 1}$, where *op* and *ip* are, respectively, the output and input sizes of the layer. The output of this fully connected layer w.r.t an input $i \in R^{ip \times T_s}$ will be

$$o = W_{fc} \times i + b_{fc}$$
$$where\ output\ o \in R^{op \times T_s}$$

Thus, we can see that the layer functionality does not alter the output size for a variable length of time sequence, making the functionality of this layer independent of the time series length.
The reachability of a fully-connected layer will be given by the following lemma.

Lemma 1. *The reachable set of a fully-connected layer with a Star input set* $I = \langle c, V, P \rangle$ *is another Star* $I' = \langle c', V', P' \rangle$ *where* $c' = W_{fc} \times c + b_{fc}$*, the matrix multiplication of c with Weight matrix* W_{fc}*,*$V' = \{v'_1, ..., v'_m\}$*, where* $v'_i = W_{fc} \times v_i$*, the matrix multiplication of the weight matrix and the* i^{th} *basis vector, and* $P' = P$*.*

Reachability of a 1D Convolutional Layer. We consider a 1d convolution layer with the following parameters: the weights $W_{conv1d} \in R^{w_f \times nc \times fl}$ and the bias $b_{conv1d} \in R^{1 \times fl}$ where $w_f, nc\ and\ fl$ are the filter size, number of channels and number of filters, respectively.

The output of this 1d convolution layer w.r.t an input $i \in R^{ip \times T_s}$ will be

$$o = W'_{conv1d} \cdot i' + b_{conv1d}\ \ dot\ product\ along\ time\ dimesion\ for\ each\ filter$$
$$where\ output\ o \in R^{fl \times T'_s}$$

where $T'_s = T_s + T_d - T_{fl}$ is the new time series length at the output and T_d, T_{fl} are the time lengths contributed by the dilation factor and the 1d convolution

function, respectively. w'_{conv1d} is the modified weight matrix after adding dilation, and i' is the modified input after padding. We can see when T_d becomes equal to T_{fl} for any convolution layer, the layer functionality becomes independent of the length of the time series.
The reachability of a 1d convolution layer will be given by the following lemma.

Lemma 2. *The reachable set of a 1d convolution layer with a Star input set* $I = \langle c, V, P\rangle$ *is another Star* $I' = \langle c', V', P'\rangle$ *where* $c' = W_{conv1d} \cdot c$*, 1d convolution applied to the basis vector c with Weight matrix* W_{conv1d},$V' = \{v'_1, ..., v'_m\}$*, where* $v'_i = W_{conv1d} \cdot v_i$*, is the 1d convolution operation with zero bias applied to the generator vectors, i.e., only using the weights of the layer, and* $P' = P$*.*

7 Experimental Setup

7.1 Dataset Description

For evaluation, we have considered two different time series datasets for PHM of a Li battery and a turbine.

Battery State-of-Charge Dataset (BSOC) *[14]:* This dataset is derived from a new 3Ah LG HG2 cell tested in an 8 cu.ft. thermal chamber using a 75amp, 5-volt Digatron Firing Circuits Universal Battery Tester with high accuracy (0.1 of full scale) for voltage and current measurements. The main focus is to determine the State of Charge (SOC) of the battery, measured as a percentage, which indicates the charge level relative to its capacity. SOC for a Li-ion battery depends on various features, including voltage, current, temperature, and average voltage and current. The data is obtained from the 'LG_HG2_Prepared_Dataset_McMasterUniversity_Jan_2020', readily available in the dataset folder [14]. The training data consists of a single sequence of experimental data collected while the battery-powered electric vehicle during a driving cycle at an external temperature of 25 degrees Celsius. The test dataset contains experimental data with an external temperature of -10 degrees Celsius.

Turbofan Engine Degradation Simulation Data Set (TEDS) *[2, 30]:* This dataset is widely used for predicting the Remaining Useful Life (RUL) of turbofan jet engines [2]. Engine degradation simulations are conducted using C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) with four different sets, simulating various operational conditions and fault modes. Each engine has 26 different feature values recorded at different time instances. To streamline computation, features with low variability (similar to Principal Component Analysis [26]) are removed to avoid negative impacts on the training process. The remaining 17 features [A.5, A.4] are then normalized using z-score (mean-standard deviation) for training. The training subset comprises time series data for 100 engines, but for this paper, we focus on data from only one engine (FD001). For evaluation, we randomly selected engine 52 from the test dataset.

7.2 Network Description

The network architecture used for training the BSOC dataset, partially adopted from [1], is a regression CNN (as shown in [Fig 9, A.7]). The network has five input features which correspond to one SOC value. Therefore, the TSRegNN for the BSOC dataset can be represented as:

$$\begin{aligned} f: \ x \in \mathbb{R}^{5\times t_s} &\rightarrow y \in \mathbb{R}^{1\times t_s} \\ S\hat{O}C_{t_s} &= f(t_s) \end{aligned} \tag{17}$$

The network architecture used for training the TEDS dataset is also a regression CNN, adopted from [3] (shown in [Fig 9, A.7]). The input data is preprocessed to focus on 17 features, corresponding to one RUL value for the engine. Therefore, the TSRegNN for the TEDS dataset can be represented as:

$$\begin{aligned} f: \ x \in \mathbb{R}^{17\times t_s} &\rightarrow y \in \mathbb{R}^{1\times t_s} \\ R\hat{U}L_{t_s+1} &= f(t_s) \end{aligned} \tag{18}$$

The output's t_s^{th} value represents the desired estimation of SOC or RUL, with the given series of past t_s values for each feature variable.

8 Experimental Results and Evaluation

The actual experimental results shown in this paper are conducted in a Windows-10 computer with the 64-bit operating system, Intel(R) Core(TM) i7-8850H processor, and 16 GB RAM.

For all four noise scenarios [Sec. 3], local and global (for 100 consecutive time steps) robustness properties are considered for both datasets. The local monotonicity property is only considered for the turbine RUL estimation example.

Battery State-of-Charge Dataset (BSOC): In this dataset, the output value (SOC) is supposed to be any value between 0 and 1 (or 0 and 100%). But, for the instances where the lower bound is negative, we instead treat it as 0 because a negative SOC does not provide any meaningful implications.

For SFSI, for a random (here feature 3) input feature-signal, the noise is added only at the last time step (t_{30}) of the 3rd feature, whereas for SFAI, noise is added throughout all the time instances of the input signal. The effect of four different noise values, 1%, 2.5%, 5% and 10% of the mean(μ), are then evaluated using over-approximate star reachability analysis [Sec. 2.3] on 100 consecutive input signal, each with 30 time instances. We considered $\pm$5% around the actual SOC value as the allowable bounds. For all the noises, 2 different robustness values, PR [Def. 7] and POR [Def. 8] are then calculated, and comparative tables are shown below in Table 1.

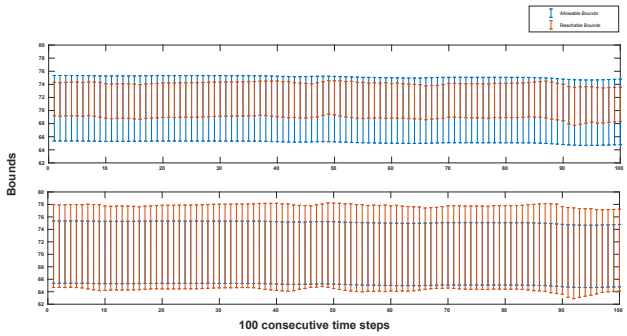

Fig. 2: Allowable (blue) and reachable (red) bounds for battery SOC dataset for 100 consecutive time steps and 2 different SFAI noise values 1% (upper), and 2.5% (lower) respectively

Table 1: Global Robustness: Percentage Robustness(PR) for noises for 100 consecutive time steps

$noise$	PR_{SFSI}	POR_{SFSI}	$avgRT_{SFSI}(s)$	PR_{SFAI}	POR_{SFAI}	$avgRT_{SFAI}(s)$
1	100	100	0.7080	100	100	20.9268
2.5	100	100	0.7080	100	100	20.9991
5	100	100	0.7116	100	100	21.0729
10	100	100	0.7027	100	100	21.0780

$noise$	PR_{MFSI}	POR_{MFSI}	$avgRT_{MFSI}(s)$	PR_{MFAI}	POR_{MFAI}	$avgRT_{MFAI}(s)$
1	100	100	0.7653	100	100	36.1723
2.5	0	73.87	0.8251	0	73.87	59.0588
5	0	35.95	0.9026	0	35.95	91.6481
10	0	17.89	1.1051	0	17.89	163.7568

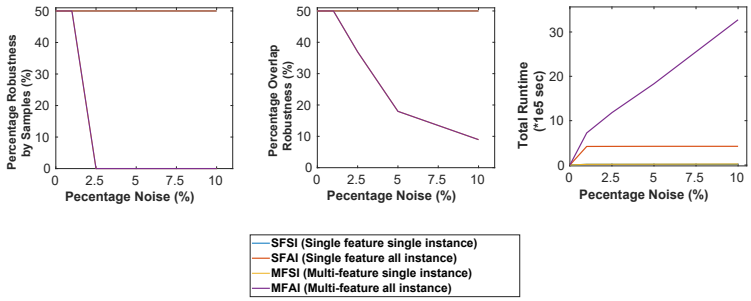

Fig. 3: Percentage Robustness and Runtime plots w.r.t increasing noise

Observation and Analysis: Fig. 2 shows a sample plot for gradually increasing estimation bounds with increasing MFSI noise. We can see from the figure that for each time instance, the system becomes locally non-robust as the noise value increases.

Table. 1 presents the network's overall performance, i.e., the percentage robustness measures, PR [Def. 7], POR [Def. 8] and average verification runtime (avgRT), with respect to each noise. The percentage robustness values start decreasing and the average (as well as total) runtime starts increasing as the measure of noise increases for MFAI and MFSI, but for SFSI and SFAI it remains the same for these noise perturbations considered. This is because in the first case, the noise is added to all the features, resulting in increasing the cumulative effect of disturbance on the output estimation. However, in the other case, the noise is attached only to a single feature, assuming that not all features will get polluted by noise simultaneously; and that the reachable bounds are in the acceptable range. A plot of robustness values and the total runtime is shown in Fig 3.

We can also see that the decrease in POR values for MFSI and MFAI are less compared to the PR values with increasing noise because, for PR calculation, only those time steps are considered where the estimated range falls entirely within the allowed range, whereas for POR calculation even if some part of the estimated range goes outside the allowable range, their fractional contribution is still considered.

Another interesting observation here is the robustness matrices for both SFSI and SFAI are the same; however, the computations for SFAI take almost three times longer than the computations for SFSI. The same analogy is observed for MFSI and MFAI datasets but with an even higher time taken for MFAI. The possible reason for this observation could be that, while the data is subjected to perturbations across all time instances, the noise added to the final time step has the most significant impact on the output.

Turbofan Engine Degradation Simulation Data Set (TEDS): In this dataset, the acceptable RUL bounds are considered to be ± 10 of the actual RUL. For instances where the lower bound is negative, we assume those values to be 0 as well. We then calculate the percentage robustness measures, PR [Def.7], POR [Def.8], and average verification runtime (avgRT), for an input set with all 100 consecutive data points, each having 30 time instances. The results for three different noise values, 0.1%, 0.5%, and 1% of the mean (μ), are presented in Table 2. For SFSI and SFAI noises, we randomly choose a feature (feature 7, representing sensor 2) for noise addition. The noise is added to the last time step (t_{30}) of each data sample for SFSI and SFAI noises. The results of the MFAI noise have been omitted due to scalability issues, as it is computationally heavy and time-consuming [3].

For verifying the local monotonicity of the estimated output RUL bounds at a particular time instance, we have fitted the previous RUL bounds along with

[3] The MFAI noise, i.e., adding the L_∞ norm to all feature values across all time instances, significantly increases the input-set size compared to other noise types. This leads to computationally expensive calculations for layer-wise reachability, resulting in longer run times. Moreover, noise in an industrial setting affecting all features over an extended period is unlikely. Considering these factors, we decided to exclude the results of the MFAI noise for the TEDS dataset from our analysis.

the estimated one in a linear equation as shown in Fig. 4. This guarantees the monotonically decreasing nature of the estimated RUL at any time instance.

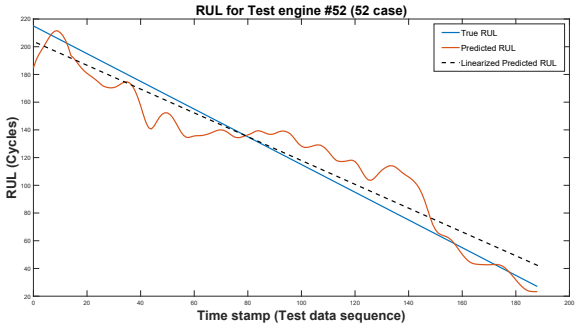

Fig. 4: Percentage Robustness and Runtime plots w.r.t increasing noise

Table 2: Global Robustness: Percentage Robustness(PR) for noises for 100 consecutive time steps

$noise$	PR_{SFSI}	POR_{SFSI}	$avgRT_{SFSI}(s)$	PR_{SFAI}	POR_{SFAI}	$avgRT_{SFAI}(s)$
1	13	13	1.0796	13	13.31	32.8670
2.5	13	13	1.1755	12	13.13	62.1483
5	13	13	1.2908	8	12.64	108.0736

$noise$	PR_{MFSI}	POR_{MFSI}	$avgRT_{MFSI}(s)$
1	13	13	9.6567
2.5	13	13	10.2540
5	13	13	11.2100

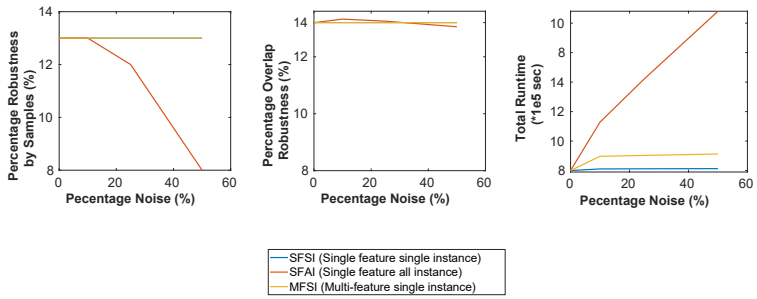

Fig. 5: Percentage Robustness and Runtime plots w.r.t increasing noise

Observation and Analysis: Fig. 14 of the detailed version of this paper [25] shows a sample plot for gradually increasing estimation bounds with increasing SFAI noise. Here we need to notice that the network's performance in terms of following the actual RUL value is not well. However, Table. 2 presents the

network's overall performance with respect to each noise. Contrary to the other dataset, we see that the percentage robustness measures corresponding to SFAI and SFSI noises differ. Interestingly, while the noise value increases, the PR, and POR for SFSI remain the same, whereas the robustness measures for SFAI decrease. However, the performance matrices for MFSI are the same as the SFSI except for the time. This might be because, for both SFSI and MFSI, the noise is added only at a single time instance, whereas for SFAI, the noise is added to the entire time instances, resulting in an increased cumulative effect of disturbance on the output.

Our results consistently show higher POR values than PR values in Table. [1-2]. Since we assess output reachable bounds using L_∞ perturbations in the input, we acknowledge the significance of cases where reachable sets overlap with permissible bounds but do not entirely fall within them. In summary, PR measures adopt a more conservative approach, while POR captures the relationship between output reachable bounds and permissible bounds more accurately.

9 Conclusion and Future Work

This paper explores formal method-based reachability analysis of variable-length time series regression neural networks (NNs) using approximate Star methods in the context of predictive maintenance, which is crucial with the rise of Industry 4.0 and the Internet of Things. The analysis considers sensor noise introduced in the data. Evaluation is conducted on two datasets, employing a unified reachability analysis that handles varying features and variable time sequence lengths while analyzing the output with acceptable upper and lower bounds. Robustness and monotonicity properties are verified for the TEDS dataset. Real-world datasets are used, but further research is needed to establish stronger connections between practical industrial problems and performance metrics. The study opens new avenues for exploring perturbation contributions to the output and extending reachability analysis to 3-dimensional time series data like videos. Future work involves verifying global monotonicity properties as well, and including more predictive maintenance and anomaly detection applications as case studies. The study focuses solely on offline data analysis and lacks considerations for real-time stream processing and memory constraints, which present fascinating avenues for future research.

Acknowledgements. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1910017, 2028001, 2220426, and 2220401, and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089 and FA8750-23-C-0518, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF. We also want to thank our colleagues, Tianshu and Bernie for their valuable feedback.

References

1. Predict Battery State of Charge Using Deep Learning - MATLAB &; Simulink — mathworks.com. https://www.mathworks.com/help/deeplearning/ug/predict-soc-using-deep-learning.html
2. Prognostics center of excellence - data repository. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan
3. Remaining Useful Life Estimation Using Convolutional Neural Network - MATLAB &; Simulink — mathworks.com. https://www.mathworks.com/help/predmaint/ug/remaining-useful-life-estimation-using-convolutional-neural-network.html
4. Anderson, G., Pailoor, S., Dillig, I., Chaudhuri, S.: Optimization and abstraction: a synergistic approach for analyzing neural network robustness. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 731–744 (2019)
5. Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., Schilling, C.: Juliareach: a toolbox for set-based reachability. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 39–44 (2019)
6. Borgi, T., Hidri, A., Neef, B., Naceur, M.S.: Data analytics for predictive maintenance of industrial robots. In: 2017 International Conference on Advanced Systems and Electric Technologies (IC_ASET). pp. 412–417. IEEE (2017)
7. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., Misener, R.: Efficient verification of relu-based neural networks via dependency analysis. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 3291–3299 (2020)
8. DeLillo, D.: White noise. Penguin (1999)
9. EASA, Aerospace, C.: Formal methods use for learning assurance (formula). Tech. rep. (4 2023)
10. Ferguson, C.E.: Time-series production functions and technological progress in american manufacturing industry. Journal of Political Economy **73**(2), 135–147 (1965)
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
12. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification. pp. 443–452. Springer (2019)
13. Kauffman, S., Dunne, M., Gracioli, G., Khan, W., Benann, N., Fischmeister, S.: Palisade: A framework for anomaly detection in embedded systems. Journal of Systems Architecture **113**, 101876 (2021)
14. Kollmeyer, P., Vidal, C., Naguib, M., Skells, M.: Lg 18650hg2 li-ion battery data and example deep neural network xev soc estimator script. Mendeley Data **3**, 2020 (2020)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
16. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks **8**(1), 98–113 (1997)
17. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

18. Lin, C.Y., Hsieh, Y.M., Cheng, F.T., Huang, H.C., Adnan, M.: Time series prediction algorithm for intelligent predictive maintenance. IEEE Robotics and Automation Letters **4**(3), 2807–2814 (2019)
19. Lopez, D.M., Choi, S.W., Tran, H.D., Johnson, T.T.: Nnv 2.0: The neural network verification tool. In: International Conference on Computer Aided Verification. pp. 397–412. Springer (2023)
20. Lv, F., Wen, C., Liu, M., Bao, Z.: Weighted time series fault diagnosis based on a stacked sparse autoencoder. Journal of Chemometrics **31**(9), e2912 (2017)
21. Martinez, C.M., Cao, D.: iHorizon-Enabled Energy management for electrified vehicles. Butterworth-Heinemann (2018)
22. Mohapatra, J., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Towards verifying robustness of neural networks against a family of semantic perturbations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 244–252 (2020)
23. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
24. Müller, M.N., Brix, C., Bak, S., Liu, C., Johnson, T.T.: The third international verification of neural networks competition (vnn-comp 2022): summary and results. arXiv preprint arXiv:2212.10376 (2022)
25. Pal, N., Lopez, D.M., Johnson, T.T.: Robustness verification of deep neural networks using star-based reachability analysis with variable-length time series input. arXiv preprint arXiv:2307.13907 (2023)
26. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science **2**(11), 559–572 (1901)
27. Priemer, R.: Introductory signal processing, vol. 6. World Scientific (1991)
28. Priemer, R.: Signals and signal processing. Introductory Signal Processing pp. 1–9 (1991)
29. de Riberolles, T., Zou, Y., Silvestre, G., Lochin, E., Song, J.: Anomaly detection for ics based on deep learning: a use case for aeronautical radar data. Annals of Telecommunications pp. 1–13 (2022)
30. Saxena, A., Goebel, K.: Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository pp. 1551–3203 (2008)
31. Semenick Alam, I.M., Sickles, R.C.: Time series analysis of deregulatory dynamics and technical efficiency: the case of the us airline industry. International Economic Review **41**(1), 203–218 (2000)
32. Sivaraman, A., Farnadi, G., Millstein, T., Van den Broeck, G.: Counterexample-guided learning of monotonic neural networks. Advances in Neural Information Processing Systems **33**, 11936–11948 (2020)
33. Soomro, K., Bhutta, M.N.M., Khan, Z., Tahir, M.A.: Smart city big data analytics: An advanced review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **9**(5), e1319 (2019)
34. Stübinger, J., Schneider, L.: Understanding smart city—a data-driven literature review. Sustainability **12**(20), 8460 (2020)
35. Susto, G.A., Beghi, A.: Dealing with time-series data in predictive maintenance problems. In: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 1–4. IEEE (2016)
36. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)

37. Touloumi, G., Atkinson, R., Tertre, A.L., Samoli, E., Schwartz, J., Schindler, C., Vonk, J.M., Rossi, G., Saez, M., Rabszenko, D., et al.: Analysis of health outcome time series data in epidemiological studies. Environmetrics: The official journal of the International Environmetrics Society **15**(2), 101–117 (2004)
38. Tran, H.D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using imagestars. In: International Conference on Computer Aided Verification. pp. 18–42. Springer (2020)
39. Tran, H.D., Lopez, D.M., Musau, P., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Star-based reachability analysis of deep neural networks. In: International Symposium on Formal Methods. pp. 670–686. Springer (2019)
40. Tran, H.D., Musau, P., Lopez, D.M., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Star-based reachability analsysis for deep neural networks. In: 23rd International Symposisum on Formal Methods (FM'19). Springer International Publishing (October 2019)
41. Tran, H.D., Pal, N., Musau, P., Lopez, D.M., Hamilton, N., Yang, X., Bak, S., Johnson, T.T.: Robustness verification of semantic segmentation neural networks using relaxed reachability. In: International Conference on Computer Aided Verification. pp. 263–286. Springer (2021)
42. Tran, H.D., Yang, X., Lopez, D.M., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: International Conference on Computer Aided Verification. pp. 3–17. Springer (2020)
43. Truax, B.: Handbook for acoustic ecology. Cambridge Street Records (1999)
44. Wang, Z., Wang, Y., Fu, F., Jiao, R., Huang, C., Li, W., Zhu, Q.: A tool for neural network global robustness certification and training. arXiv preprint arXiv:2208.07289 (2022)
45. Zeger, S.L., Irizarry, R., Peng, R.D.: On time series analysis of public health and biomedical data. Annu. Rev. Public Health **27**, 57–79 (2006)
46. Zhang, Z., Lai, X., Wu, M., Chen, L., Lu, C., Du, S.: Fault diagnosis based on feature clustering of time series data for loss and kick of drilling process. Journal of Process Control **102**, 24–33 (2021)

A Appendix

A.1 Preliminaries

Definition 9 (Soundness). *Let $\mathcal{F} : R^j \rightarrow R^p$ be a NN with an input set R_0 and output reachable set R_f . The computed R_f given $\mathcal{F}$ and R_0 is sound iff $\forall x \in R_0, |y = \mathcal{F}(x), y \in R_f$.*

Definition 10 (Completeness). *Let $\mathcal{F} : R^j \rightarrow R^p$ be a NN with an input set R_0 and output reachable set R_f . The computed R_f given $\mathcal{F}$ and R_0 is complete iff $\forall x \in R_0, \exists y = \mathcal{F}(x) \mid y \in R_f$ and $\forall y \in R_f, \exists x \in R_0 \mid y = \mathcal{F}(x)$.*

L_∞ Norm: Input perturbations can be quantified using different types of norms. Here, we have used the L_∞ norm, which records the greatest perturbation magnitude among all input elements.

$$L_\infty : ||x - x'||_\infty = max||x_i - x_i'|| \tag{19}$$

Example of Robustness Calculations: The left picture of Fig. 6 in Appendix depicts an example plot of output estimations (red) vs. the allowable bounds (blue). Here, we can see that the network is locally robust for time instances t_1 and t_5; in other instances, it is non-robust w.r.t the noise added. So the RV is 1 for both t_1 and t_5 and 0 for others. To better understand the concept of POR

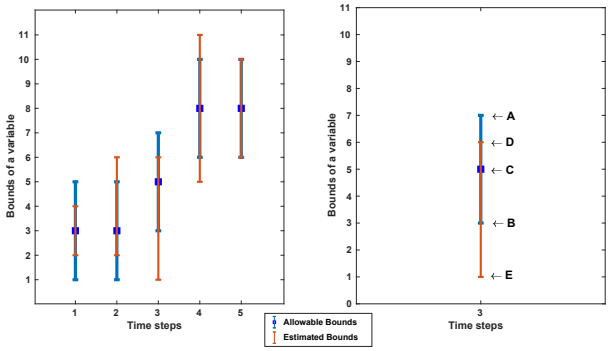

Fig. 6: Example: (left)Output estimation bounds (red) of a TSRegNN and Allowable (blue) over five consecutive time step and (right) for a particular time step t_3

and PO, we refer to the right picture of Fig. 6. Here for a time instance t_3, C denotes the actual signal value, AB is the allowable output range, and DE is the estimated reachable bounds. Here, $N_{robust} = 3$ and $N_{total} = 5$, so the PR for this particular example is 60%. The PO will be calculated as:

$$PO = \frac{DB}{DE} = \frac{3}{5} = 0.6$$

To calculate POR, we calculate the PO for each of the time instances:

$$POR = \frac{\frac{2}{2} + \frac{3}{4} + \frac{3}{5} + \frac{4}{6} + \frac{4}{4}}{5} \times 100\% = 80.33\%(approx)$$

A.2 Battery State-of-charge

Battery state-of-charge is a measurement of the amount of energy available in a battery at a specific point in time, expressed as a percentage. This term is often used in various applications involving battery-powered systems, e.g., electric vehicles, renewable energy storage systems, portable electronics etc. Accurate estimation of the State of Charge (SOC) of a battery is crucial for efficient battery management and ensuring the longevity of the battery. The SOC is expressed as a percentage of the full capacity of the battery.

A.3 Remaining Useful Life

The Remaining Useful Life (RUL) is a subjective estimate of the lifespan of any equipment before it requires repair or replacement. This important concept is often used in various fields, including maintenance, reliability engineering, and prognostics and health management (PHM). RUL estimation is typically based on the analysis of historical data, such as sensor measurements, degradation patterns, maintenance records, and operational conditions. Various techniques and models, including statistical methods, machine learning algorithms, and physics-based approaches, are generally used to predict the RUL.

A.4 Sample Dataset Plots

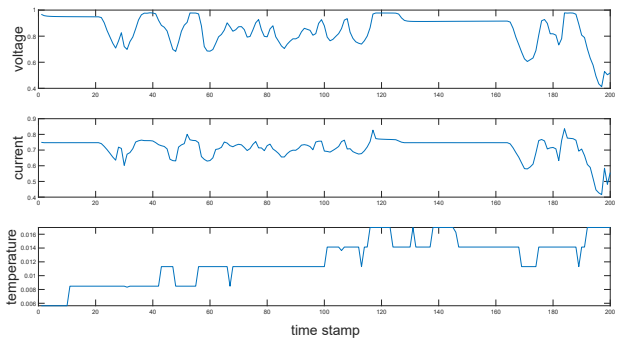

Fig. 7: Sample feature value plot for Battery SOC Dataset.

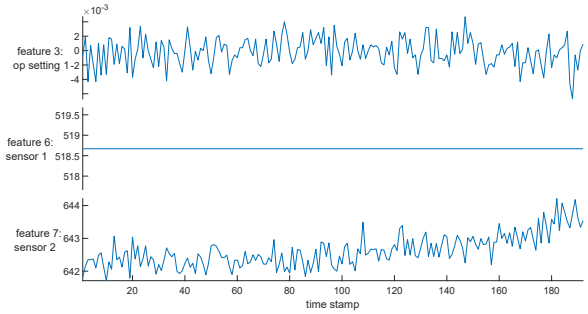

Fig. 8: Sample feature value plot for Turbofan Engine Degradation Simulation Dataset.

A.5 Prognosability

Features that remain constant for all time steps can negatively impact the training. Feature reduction is done using the 'prognosability' MATLAB command. Prognosability is actually a property relative to the prediction of the future state of the system, and this term is mainly used for lifetime data. In MATLAB, 'prognosability' is used as a function to measure the variability of the features in a dataset at failure. The equation for the prognosability calculation is given as below:

$$prognosability = Y = \exp \frac{std_j(x_j(N_j))}{mean_j|(x_j(1) - x_j(N_j)|} \tag{20}$$

The output has 3 different outcomes:

1. Y = 0 means the feature values are constant, i.e., no variability in the data.
2. Y = NaN indicates the prognosability could not be calculated.
3. Y = 1 means the feature values are perfectly prognosable i.e., there is variability in the data.

A.6 Feature Reduction of TEDS Dataset

Each engine has 26 different feature values, recorded at different time instances.

1. Feature 1: Unit number
2. Feature 2: Time-stamp
3. Feature 3–5: Operational settings
4. Feature 6–26: Sensor measurements 1–21

After analyzing the dataset using 'prognosability', number of features considered for NN training reduced to 17 from 26, and they are

1. Feature 3–4 : Operational settings 1-2
2. Feature 7–9 : Sensor measurements 2-4

3. Feature 11–14 : Sensor measurements 6–9
4. Feature 16–20 : Sensor measurements 11–15
5. Feature 22 : Sensor measurements 17
6. Feature 25-26 : Sensor measurements 20-21

A.7 Network Architectures: (Figure 9)

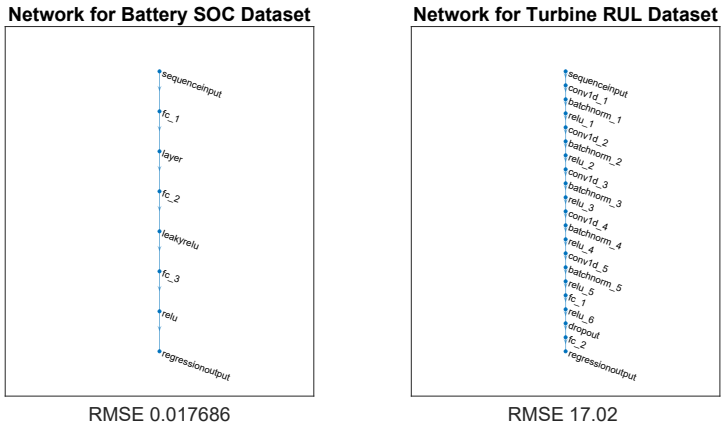

Fig. 9: The architectures of the regression networks for both the datasets.

A.8 Noise

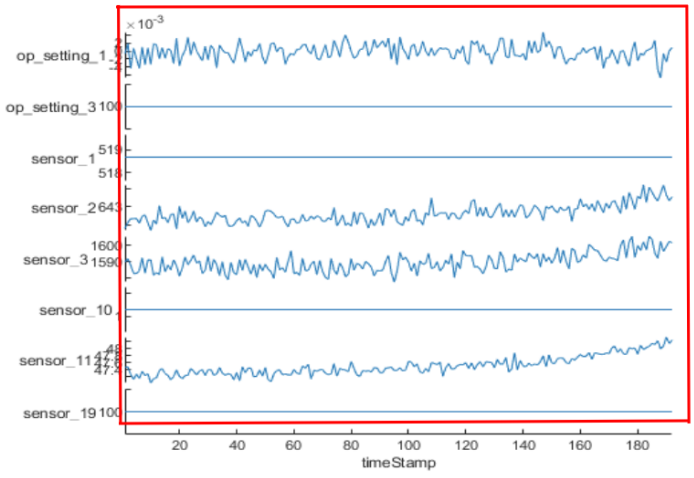

Fig. 10: MFAI Noise: noise added to the complete time series data, for all features.

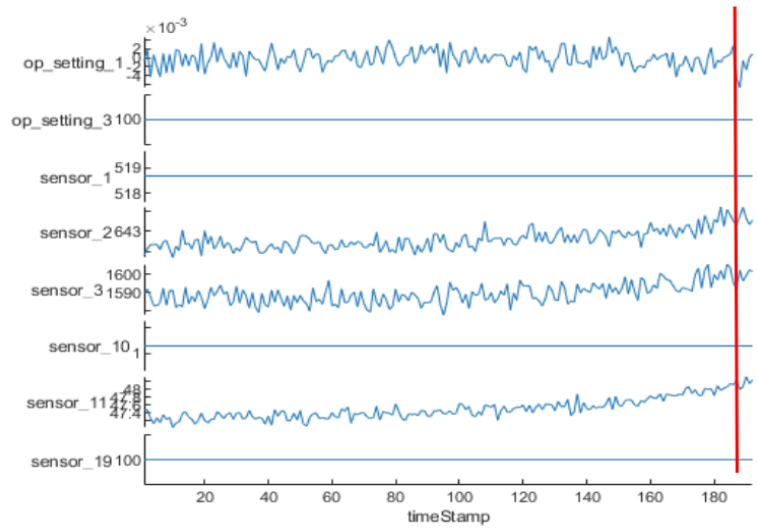

Fig. 11: MFSI Noise: noise added at a particular instance of the time series data, for all features.

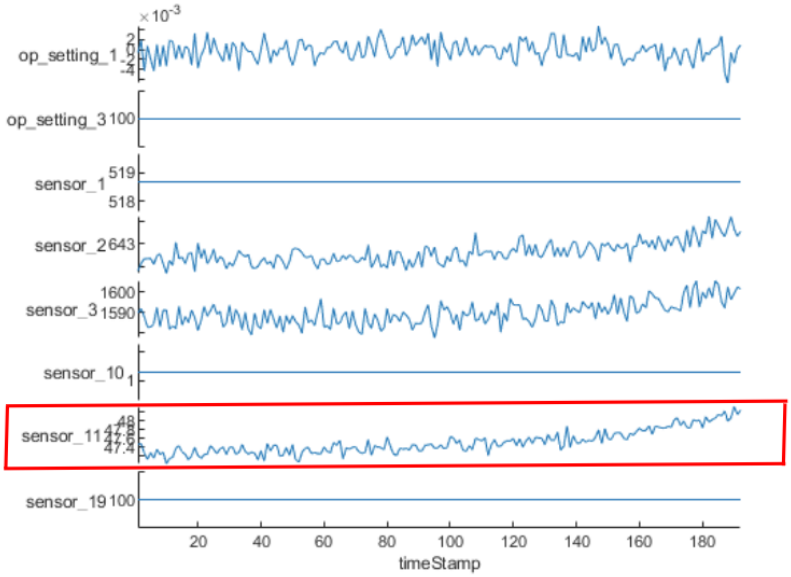

Fig. 12: SFAI Noise: noise added to the complete time series data, a particular feature.

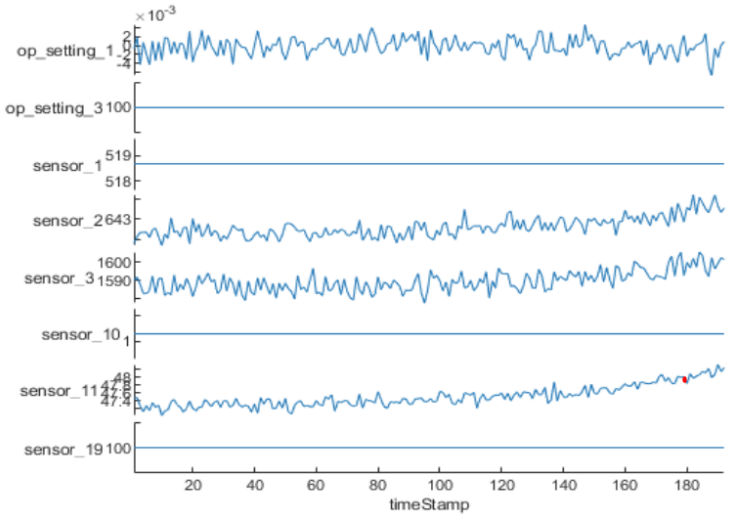

Fig. 13: SFSI Noise: noise added at a particular instance of the time series data, a particular feature.

A.9 TEDS dataset Sample Reachability Plot

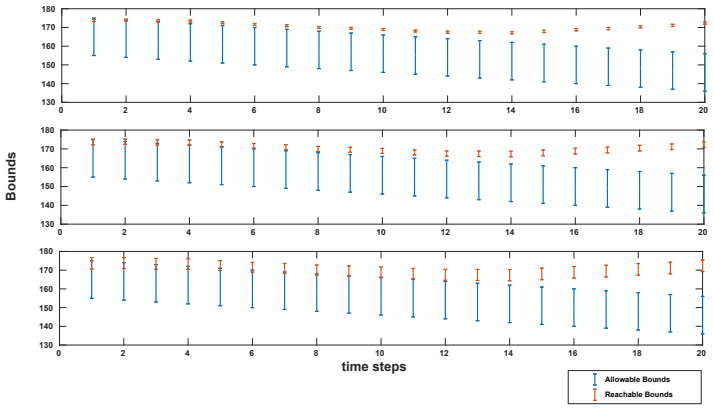

Fig. 14: Allowable (blue) and reachable (red) bounds for battery SOC dataset for 100 consecutive time steps and three different SFAI noise values 1% (upper), 2.5% (middle) and 5% (lower) respectively