TIAGO GASPAR DA ROSA

A fault prognostic framework for complex engineering systems using autoencoder type deep learning techniques

São Paulo
2023

TIAGO GASPAR DA ROSA

A fault prognostic framework for complex engineering systems using autoencoder type deep learning techniques

Dissertation presented to Escola Politécnica da Universidade de São Paulo to obtain the degree of Master of Science.

São Paulo
2023

TIAGO GASPAR DA ROSA

A fault prognostic framework for complex engineering systems using autoencoder type deep learning techniques

Revised Version

Dissertation presented to Escola Politécnica da Universidade de São Paulo to obtain the degree of Master of Science.

Concentration Area:

Mechanical Engineering of Design and Manufacturing

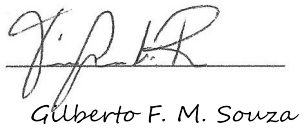
Supervisor:

Gilberto Francisco Martha de Souza

São Paulo
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 15 de Agosto de 2023

Assinatura do autor: 

Assinatura do orientador: Gilberto F. M. Souza

Catalogação-na-publicação

Gaspar da Rosa, Tiago
A fault prognostic framework for complex engineering systems using autoencoder type deep learning techniques / T. Gaspar da Rosa -- versão corr. -- São Paulo, 2023.
116 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

1.Autoencoders 2.Prognóstico de falhas 3.Redes neurais profundas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecânica II.t.

ACKNOWLEDGMENTS

Every academic work dares to navigate into the unknown to expand the frontiers of knowledge. Certain vessels can not be manned alone, so I would like to thank the USP team of the USP-COPEL project, made up of friends Fabio Norikazu Kashiwagi, Arthur Henrique de Andrade Melani and Fabio Henrique Pereira who shared with me the tasks of this long project and the challenges of conducting such work in times of remote coexistence and change.

I also thank my supervisor Prof. Ph.D. Gilberto Francisco Martha de Souza, firstly for the opportunity, as well as for the advice provided for the production of this dissertation and the suggestions that were lovingly incorporated into the body of the text.

The judging committee composed of professors outside the PPGEM Fabio Henrique Pereira and Paulo Sergio Cugnasca, who brought relevant contributions to this work and which will certainly be incorporated into the final manuscript.

The Agência Nacional de Energia Elétrica (ANEEL) and company Copel S.A. for the financing guaranteed from the R&D project PD-06491-0341-2014 “Methodology for asset management applied to hydro-generators based on mathematical models of reliability and maintainability”.

Personally, I thank Eng. M.Sc. Gisele Maria De Oliveira Salles for managing this R&D project, for making the idea credible, and for providing us with the resources we needed. And to the specialized technician Josué Carlo Betemps Vaz Da Silva, technical manager of UHE Governador José Richa, for providing assistance in technical matters and welcoming our team with open arms.

The company Compass, especially the developer Luis Gustavo da Silva, who will enable a software solution applied to the industry with the results shown here.

And finally, to the family for the support and understanding offered during the development of this work.

RESUMO

Alguns dos métodos recentes para calcular a vida útil remanescente (RUL) no contexto da manutenção baseada em condição usam o aprendizado profundo. Tal solução destaca-se pela capacidade de identificar e prever a condição dos equipamentos através da análise de grandes bases de dados. Apesar de promissor, ainda é um desafio integrar tal técnica à realidade industrial devido às restrições impostas por esses ambientes, como a tendência de mudança constante do estado operacional e a falta de detalhamento sobre os modos de falha. O presente trabalho investiga se o uso de métodos de aprendizado profundo, do tipo autoencoder, embutidos em frameworks de detecção e prognóstico projetados para fornecer previsões de um tempo de vida remanescente de forma semi-supervisionada, constituem uma solução viável para o gerenciamento da saúde de um sistema de engenharia complexo. O framework de detecção e prognóstico proposto foi testado em um exemplo de aplicação utilizando dados públicos de falhas simuladas em turbinas a jato disponibilizadas pela NASA. O método fez estimativas de RUL por meio de autoencoders treinados em condições normais de operação. Inicialmente, os dados são selecionados e pré-processados, em seguida, os autoencoders profundos são treinados em dados de condição operacional normal e um resíduo chamado erro de reconstrução de sinal é avaliado. O erro de reconstrução é pós-processado e um limite de detecção de falha é definido por canal. A estrutura é alimentada com novas entradas e quando amostras consecutivas ultrapassam o limite, a detecção ocorre e um intervalo de estado degradado é montado. A estimativa RUL é desenvolvida usando as amostras rotuladas de forma anormal por meio de um algoritmo especial com funções de decisão que destacam as tendências. Os padrões de degradação são extrapolados por um conjunto de funções univariadas até que um limiar de prognóstico, previamente determinado, sinalize a ocorrência da falta, produzindo a previsão de tempo de vida. No exemplo de aplicação, que visa confirmar a eficácia do autoencoder, três modelos são gerados com conjuntos fixos de hiperparâmetros e redes neurais e um modelo de linha de base. Depois disso, um experimento de busca em grade produz uma grande coleção de modelos com diferentes combinações de hiperparâmetros com o intuito de fazer uma análise de sensibilidade sobre o espaço de hiperparâmetros e verificar a capacidade de tuning. Os resultados permitem inferir que o framework desenvolvido é capaz de equiparar o desempenho com um modelo base que utiliza regressão linear simples em sinais pré-processados, mesmo que não supere outros modelos supervisionados apresentados na literatura. Vale ressaltar que ainda há espaço para otimização de desempenho realizando modulação de hiperparâmetros e outros elementos da arquitetura formulada. No experimento de busca em grade, a resposta pode ser mapeada para a grande maioria dos hiperparâmetros. Em seguida, os modelos foram ranqueados por meio de um método de decisão multicritério, confirmando assim a capacidade de tuning.

Palavras-Chave – Autoencoders, prognóstico, redes neurais profundas.

ABSTRACT

Some of the recent methods to calculate the remaining useful life (RUL) in the context of condition monitoring based maintenance use deep learning. It stands out as a solution capable of identifying and predicting the condition of the equipment through large databases. Although promising, it is still a challenge to integrate such a technique into industrial environments due to restrictions imposed, such as the tendency of constant change of operational state and the lack of detail about the failure modes. The present work investigates whether the use of a deep learning method of the autoencoder type, embedded in detection and prognostic framework designed to provide predictions of a remaining lifetime in a semi-supervised way, constitute a feasible solution for the health management of a complex engineering system. The proposed detection and prognosis framework was tested in an application example using public data of simulated faults in jet turbine engines made available by NASA. The method made RUL estimations through deep autoencoders trained in normal operational conditions. Initially, the data are selected and preprocessed, then the deep autoencoders are trained in normal operational condition data, and a residual called signal reconstruction error is evaluated. The reconstruction error is post-processed and a fault detection threshold is defined per channel. The framework is fed with new input and, when consecutive samples surpass the limit, the detection happens and a degraded state interval is assembled. The RUL estimation is developed using the abnormally labeled samples through a special algorithm with decision functions that highlight trends. They are extrapolated by a set of univariate functions until a prognostic threshold, previously determined, which signalizes the occurrence of the fault, producing the lifetime prediction. In the application example, which aims to confirm the autoencoder's effectiveness, three models are generated with fixed sets of hyperparameters and neural networks, and one baseline model. After that, a grid-search experiment produces a large collection of models with different combinations of hyperparameters. It is intended to run a sensitivity analysis over the hyperparameter space and to verify tuning capacity. The results allow to infer that the developed framework is capable of matching the performance with a baseline model that uses simple linear regression on pre-processed signals, even if it does not surpass other supervised models present in the literature. It is noteworthy that there are still opportunities for performance optimization by performing hyperparameter modulation and other elements of the formulated architecture. In the grid-search experiment, the response could be mapped to the vast majority of hyperparameters. Then, the models have been ranked through a multi-criteria decision method, thus confirming the capacity of tuning.

Keywords – Autoencoders, prognosis, deep neural networks.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AE	Autoencoder
AC	Application condition
CBM	Condition monitoring based maintenance
CES	Complex engineering systems
CM	Condition monitoring
CMAPSS	Commercial modular aero-propulsion system simulation
CNN	Convolutional neural network
DBM	Deep Boltzmann machine
DBN	Deep belief network
DCAE	Denoising convolutional autoencoder
DL	Deep learning
DNN	Deep neural networks
ETTF	Estimate of time to failure
GRU	Gated recurrent unit
HI	Health index
HPC	High pressure compressor
HPT	High pressure turbine
HS	Health stage
IoT	Internet of things
LPC	Low pressure compressor
LPT	Low pressure turbine
LSTM	Long short-term memory
MCDA	Multiple-criteria decision analysis
ML	Machine learning
MLP	Multilayer perceptron
MSE	Mean squared error
NLP	Natural language processing
NOC	Normal operational condition
PCA	Principal component analysis
PDF	Probability density function
PHM	Prognostics and health management
RA	Relative accuracy
RBM	Restricted Boltzmann machine
RE	Reconstruction error
RMS	Root mean square
RMSE	Root mean squared error
RTF	Run to failure
RUL	Remaining useful life
STFT	Short-time Fourier transform
TRA	Throttle-resolver angle
VAE	Variational autoencoder

LIST OF SYMBOLS

$(\cdot)^*$	Estimation
$\alpha - \lambda$	Binary metric that evaluates whether the prediction falls in an α-bounded interval centered in the actual RUL value at a specific time instance
$\Delta^{(k)}$	Difference between the predicted and the real remaining life
Δ_{sp}	Time interval between the first spotted abnormal point and the concrete tipping point for the degraded stage
γ	Fault threshold
λ	Time window modifier for $\alpha - \lambda$ *accuracy*
$\mathbf{f}$	Set of deterministic functions for prognosis
$\mathbf{I_d}$	Set of degraded state intervals
$\mathbf{p}_i$	Ordered set all time indexes before t_i
$\mathbf{r}$	Vector of residuals
$\mathbf{y_p}$	Predicted samples
$\mathbf{y_r}$	Real samples
$\mathfrak{I}_d$	Union of degraded state intervals subsets
$\pi\left[r\left(t_\lambda\right)\right]_{-\alpha}^{+\alpha}$	Probability mass of the prediction PDF
$\rho(s)$	Scalar loss function
$\sigma(PE_{th}^{(j)})$	Variariability of j^{th} channel PE_{th}
CRA_i^l	Cumulative relative accuracy
c_j	Maximum value between the post-processed RE samples labeled as NOC for the j^{th} channel
cm_{ij}	Correlation matrix entry between i and j metrics
$corr_{ij}$	Pearson correlation measure between i and j metrics
d	Detection coverage
D_{f1}	Decision function 1
D_{f2}	Decision function 2

$E(\mathbf{y_r}, \mathbf{y_p})$	Maximum error between the samples in the condition labeled as normal
$E_p(t)$	Prediction error relative to the total life of the asset
f	False-positive coverage
$F(x)$	Cost function
$f\,(t)$	PDF of the life at t
$f_{th}(t)$	Error threshold function
$H_{\top(C)}$	Prognostic horizon
I_{dc}	Discontinuity index
$inf(\cdot)$	Inferior limit
m	Number of channels
$Mon(\cdot)$	Monotonicity
N	Number of available data points
n	Subsequences size
$n(\cdot)$	Cardinality of the set
N^*	Total number of RUL estimations
$N_{int}^{(j)}$	Number of intervals where abnormality was detected in the channel j
ns	Normalized NASA´s scoring function
p	Temporal iteration step
$PE_{th}^{(j)}$	Prognostic error threshold for the j^{th} channel
$R(t)$	Survival function at t
$r^{*(t_i)}$	Remaining useful life estimation at time t_i
$R^2(\cdot)$	Coefficient of determination R squared
RA_i^l	Relative accuracy for the l^{th} prognostic experiment at time t_i
RE	Reconstruction error
$RMSE$	Root-mean-squared error
RUL_t	Random variable of the remaining useful life at time t
s	NASA´s scoring function
$s_{i,j}$	Subsequence with temporal index i, channel index j
$s_{i,j}^r$	Reconstructed subsequence mimicking $s_{i,j}$

t_d	Concrete tipping point for the degraded stage
t_f	Future time
t_{df}	Time linked to the end of the degraded state interval I_d
t_{ds}	Time linked to the start of the degraded state interval I_d
t_{eol}	Deterministic end of life time
t^*_{eol}	End of life time estimation
t_{init}	Initial time for prognosis estimations
t_{sp}	First spotted abnormal point
$w\left(r^l\right)$	Weight factor as a function of RUL
$x(t_f)$	Health condition of the machine at the future time t_f
Y_t	History of operational profiles and CM information up to t
$E[x]$	Expectancy of x
t	Current time
T_{eol}	Random variable end of life time
X_t	Random variable of the RUL
$Z(t)$	Past condition until the current time

CONTENTS

1 INTRODUCTION

In the current context of the world industry, trends are being incorporated involving the development of technologies based on the improvement of automation practices in the domain of the so-called Industry 4.0.

There has been an increase in product complexity with the rapid improvement of modern technology, while better quality and reliability are demanded, increasing the cost of preventive maintenance which has become a major expense in many industrial companies. Given this scenario, new maintenance approaches are required (JARDINE; LIN; BANJEVIC, 2006). Therefore, one of the aspects of the smart factory lies in increasing analytical sophistication from a descriptive level - what is happening - to a diagnostic level - why does it happen - and predictive - what will happen.

The incorporation of Internet of Things (IoT) into maintenance has brought new possibilities, strengthening condition monitoring based maintenance (CBM). CBM aims to avoid unnecessary maintenance tasks by taking maintenance actions only when there is evidence of abnormal behavior of physical assets. If a CBM program is properly established and effectively implemented, it can significantly reduce the cost by minimizing the number of scheduled preventive maintenance operations (JARDINE; LIN; BANJEVIC, 2006).

The main feature of CBM is the condition monitoring (CM) process, in which signals are continuously monitored from certain types of sensors or other appropriate indicators to show the current state of a system or component (AHMAD; KAMARUDDIN, 2012). According to Ahmad and Kamaruddin2012 the purpose of the CM is to initially collect the condition data (information) about the equipment and then add knowledge about the causes of failure and their respective effects and also about the degradation patterns. Thus, a CBM program consists of three key steps (AHMAD; KAMARUDDIN, 2012): data acquisition (information collection), data processing (information processing), for understanding and interpretation through analysis, and decision making aiming at recommending efficient maintenance policies.

The sequence of steps mentioned above results in two important forms of analysis within a CBM program: diagnosis and prognosis. According with Jardine, Lin and Banjevic (2006), diagnosis deals with fault detection, isolation of faulty components, and identification of abnormal occurrence , and prognosis deals with predicting the fault before it occurs. Failure prognosis is a task to determine if a failure is imminent and to estimate how soon and how likely this failure is to occur . Prognosis is much more efficient than diagnostics for achieving zero downtime performance. Diagnosis, however, is necessary when the failure prediction is not sufficient and failure occurs . A comprehensive description of prognosis and prognosis modeling is provided by ISO 13381-1 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015), which defines it as: "an estimate of the time to failure and risk for one or more existing or future failure modes".

The estimation is also recognized in the literature as remaining useful life (RUL) and is alternatively conceptualized as a portion of time from the current moment to the end of the asset's or system's useful life, thus one of the key factors in condition-based maintenance, prognostics and health management (PHM). Different types of techniques were employed over the years in order to evaluate it.

Recently, the deep learning (DL) methods have been gaining ground in the context of prognosis and health management as they are solutions capable of identifying and predicting the condition of equipment only through the analysis of large datasets. They are useful in circumstances where there is little or no investigation into the physics of the failure. However, the current state of the art with regard to industrial scale integrated solutions is still incipientor real data with artificial faults. Thereby, some challenges that must be surpassed, like the capacity of learning in environments with evolving operating conditions, novelty detection, robustness to changes in the operational conditions and the capacity of generalization and output interpretability (FINK et al., 2020).

In summary, the importance of predictive maintenance research has grown significantly in the last 25 years (Montero Jimenez et al., 2020). Anyhow the area still presents challenges to be solved in order to consolidate its application on an industrial scale, given that its scope, complexity, and interdisciplinarity have the possibility of adopting decisions in engineering processes.

1.1 Justification

Prognosis is a relatively recent research area that has being developed in the face of new technologies implemented in the industry. Within the scope of CBM, it is one of the main research topics in reliability and maintenance engineering nowadays, following its importance as a methodology to be implemented as a result of new industrial demands and the emergence of new resources to be used to supply them.

Most of the research works on prognosis have been theoretical, disparate, and restricted to a small number of models and failure modes (SIKORSKA; HODKIEWICZ; MA, 2011). According to the same authors, there are still few published examples of prediction models being applied in the field, in complex systems, exposed to an usual range of operational and business conditions.

Currently, prognostic modeling mainly emphasizes single components, even if multi-dimensional degradation data of subsystems are available. Thus, robust solutions that take advantage of complex data structures, adaptative to online data, and that consider the influence of external variables are still necessary (GUO; LI; LI, 2019).

Furthermore, a systematic methodology for implementing prognostic models based on deep learning which considers the restrictions of the system - in relation to data management, expert knowledge, or the physical behavior of the degradation process – and specificities of the business at the time of selection of prognosis models, as well as details of their combination that are expressed in optimal levels of precision and accuracy to predict RUL in relation to a failure mode was not found in the literature.

The present dissertation allows conducting investigations into limiting aspects of the use of prognosis in mechanical systems in real situations, susceptible to the inherent difficulties of their own operational complexity. Many prognosis models are designed to predict a component failure mode without considering the component's interaction with other components or the operating environment (HENG et al., 2009).

Deep learning is suitable for this problem due to its ability to the ability to automate the processing of a great among of condition monitoring data, extract useful features from high dimensional heterogeneous data sources, learn functional and temporal relationships between and within the signal time series and by attenuating the need for feature engineering in datasets composed of many monitored variables (FINK et al., 2020).

In a more specific manner, an autoencoder architecture is chosen because of its capacity to learn the normal system behavior and distinguish it from system states that

differ from those observed under the labeled condition (FINK et al., 2020). Hence, they seem to have the potential to perform anomaly detection and also can serve as support for prognosis, whether the residue is extrapolated to predict the remaining life. In both, compression capability stands out for enabling information fusion of multiple channels together with dimensionality reduction.

1.2 Objectives

The main objective is to determine whether the use of autoencoder type deep learning methods can be a feasible solution for health management to be integrated into a complex engineering system. It is designed to provide predictions of remaining useful life in an semi-supervised way. The complex system, in this context, is understood as a multicomponent, subject to the given conditions: large characteristic space from signal data, little detail regarding fault physics, or rarefied fault history, and subject to operational modes transition.

In addition, the following are secondary objectives:

- The definition of a group of metrics designed to evaluate the performance of different variations of built models, ensuring comparability between them for validation and improvement purposes
- The analysis of the behavior of the neural networks in relation to the modulation of parameters to verify the sensitivity of the response of this space on the performance, fixed to the same type of input and adaptive capacity by retraining in different input conditions. By type of similar input is meant the combination between the same normal condition, that is, systems subject to the same operational mode, with the progression of also analogous failure modes.

1.3 Dissertation Outline

The present dissertation is divided in the following way:

- Chapter 1 contains an introduction, justification, and objectives of this work.
- Chapter 2 presents a preliminary conceptualization of prognosis, provides a classification schema for prognostic models, and shows a survey about types of prognostic frameworks. It also discusses the use of deep learning in prognostics and health management, elucidating its capacity to deal with previously intractable issues in the field and to facilitate the handling of already solved ones. It displays a repertoire of traditional techniques and remembers previous applications in the recent literature. The review focuses on autoencoders, convolutional neural networks, recurrent structures, and combined approaches.
- Chapter 3 presents the method to estimate remaining useful life by application of deep autoencoders trained in normal operational conditions. Moreover, the chapter discusses the reasons for choosing this type of approach. Then describes the main steps of the prognostic framework, which includes data preparation, fault detection, fault prognosis, and performance assessment.
- Chapter 4 deliberates on an application example, in which the method is used. It documents a study conducted in a public simulated database of turbofan engines, in which degradation patterns are obtained from the input of real flight conditions. It is fractioned in two parts: initially generates three models with fixed sets of hyperparameters and neural networks and one baseline model, and aims to confirm the autoencoder effectiveness. After that, a grid-search experiment produces a large collection of models with different combinations of hyperparameters. It is intended to run a sensitivity analysis over the hyperparameter space and verify tunning capacity. The analysis has three main parts, each one in a subsection: a global analysis of the metric space response, temporal profiles for detection and prognostic activities, and finally correlation and clustering of metrics and model ranking.
- Chapter 5 encompasses the dissertation main conclusions found.

2 LITERATURE REVIEW

This chapter exposes introductory concepts related with prognostics. After that, defines and provides examples of diagnostics and prognostics frameworks. Additionally, presents a classification schema for prognostic models. Then discusses potential applications of deep-learning in health management and enumerates the most used techniques. It finalizes with a literature review focused in autoencoders, convolutional neural networks, recurrent structures and highlighted use cases.

2.1 Preliminary Conceptualization

The word prognosis originates from the Greek etymology "*prognôskein*" which means to know in advance, being a methodological proposal or an end in itself that permeates different areas of research in medicine, meteorology, engineering and economics. It has also been incorporated into industrial maintenance polices as a resource for asset management decision-making at systemic and component levels.

The last strand, in turn, is a recent research topic, with pioneering works dating from the 90s, soon earning popularity in recent years due to the emergence of new analysis techniques based on data from other fields, innovations in hardware, and increasing availability of open-source computational libraries, which facilitate model prototyping and iteration, as well as datasets pre-processing.

Although the number of publications referring to prognosis in reliability engineering and related journals is increasing, there is still no consensus among authors on a universal definition for this term. When investigating this topic, Sikorska, Hodkiewicz and Ma (2011) grouped a set of characteristics inferred from a collection of definitions presented in the literature, compiled in Table 1. The author arrived at the understanding that:

1. Prognosis is, and should be, performed at the component or sub-component level;

2. Prognosis involves predicting the temporal progression of a specific fault from its

incipience to component failure;

3. Requires an estimate or future operation of the component;

4. Strictly related to diagnosis.

It can be seen that a prognostic task provides a certain predictability for asset failure, as foreseen by item 2 above. The resulting future forecast supports the decision-making process and must be considered as subject to uncertainties, which in turn require ways to be represented and managed.

An immediate consequence is the formulation of a prognostic result, which is composed of two parts: an estimate of time to failure (ETTF) or remaining useful life (RUL) of the asset and a confidence interval resulting from the propagated uncertainty, which according to Sikorska, Hodkiewicz and Ma (2011) is implicit in the above definitions. Some authors prefer to consider the RUL a random variable that depends on the current state of the asset, operation environment and the observed condition monitoring, which is an equally valid variation of the postulate above (SI et al., 2011).

The more conventional concept is that the remaining useful life of an asset or system is defined as the interval between the current time and the end of the useful life of the asset (SI et al., 2011). The understanding of different authors follows below.

For Jardine, Lin and Banjevic (2006) it refers to the time remaining before failure observation given the current condition and age of the machine and past operating profile. The authors understands it as a conditional random variable of the form:

$$RUL_t \sim T_{eol} - t \mid T_{eol} > t, Z(t) \tag{2.1}$$

where T_{eol} indicates a random variable time to failure or end of life, t the current time, and $Z(t)$ the past condition until the current time. The estimation of remaining life means finding the distribution of the RUL or evaluating its expectation of it:

$$E[T_{eol} - t \mid T_{eol} > t, Z(t)] \tag{2.2}$$

For Si et al. (2011), given the random variable RUL_t at the time t associated with the probability density function (PDF) of RUL_t conditional on Y_t, which is denoted by the authors as $f(RUL_t|Y_t)$. Therefore:

$$f\left(RUL_t \mid Y_t\right) = f\left(RUL_t\right) = \frac{f\left(t + RUL_t\right)}{R(t)} \tag{2.3}$$

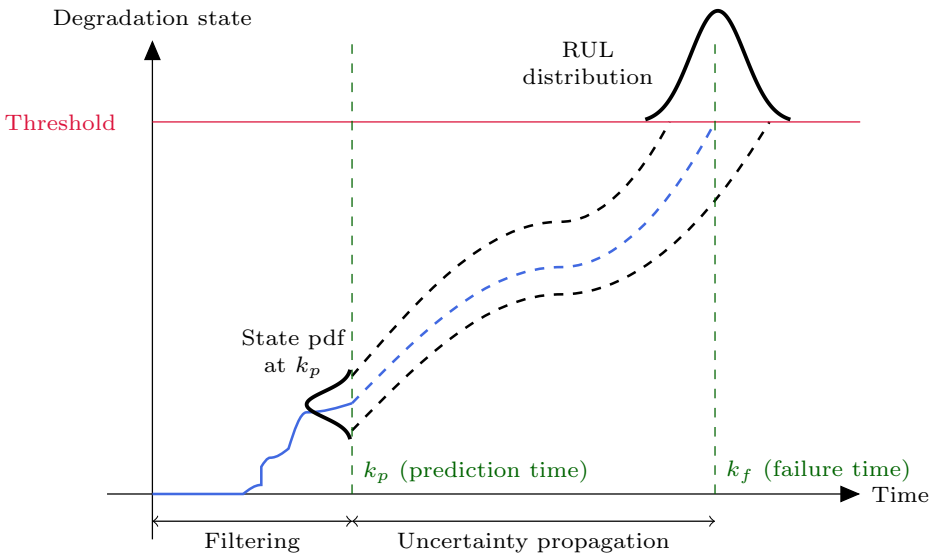
where $f\left(t+X_t\right)$ is the PDF of the life at $t+RUL_t$, equivalent to t_{eol}, $R(t)$ is the survival function at t and Y_t is the history of operational profiles and CM information up to t.

Based on the fact that many publications designate RUL as "the time left before the health state - or index - of machinery cross a fault threshold" the subsequent expression has been formulated (LEI, 2016)(LEI et al., 2018):

$$t_{eol} = \inf\left(t_f \mid x\left(t_f\right) \geqslant \gamma\right) \tag{2.4}$$

where, $r(t) = t_{eol} - t$ is the remaining life at time t, $inf(\cdot)$ is the inferior limit of a variable, $x(t_f)$ is a function that represents the health condition of the machine at the future time t_f and γ is a fault threshold. For the intent of uncertainty quantification, the fault threshold should be described as a probability distribution instead of a constant value (LEI et al., 2018). Figure 1 illustrates the concept indicating a prognosis process by extrapolating a degradation pattern until a limit. When it is exceeded, the equipment enters a fault state. In the drawing, the 2$^{\text{nd}}$ order uncertainties, resulting from the trend extrapolation, and the first order, which originated in the definition of the limit, are represented.

Figure 1: Example of a simple prognosis process by extrapolating a degradation pattern to the tolerated limit for the magnitude.



Source: Robinson (2018)

Table 1: Definitions for prognosis

Reference	Prognosis is... (Quoted Directly)
Lewis and Edwards (1997)	Prediction of when a failure may occur. To calculate the remaining useful life on an asset
Engel et al. (2000)	The capability to provide early detecting of the precursor and/or incipient fault condition of a component, and to have the technology and means to manage and predict the progression of this fault condition to component failure
Brotherton et al. (2002)	The ability to assess the current health of a part for a fixed time horizon or predict the time to failure
Luo et al. (2003)	Failure prognosis involves forecasting of system degradation based on observed system condition
Smith, Coit and Liang (2003)	The capability to provide early detection and isolation of precursor and/or incipient fault condition to a component or sub-element failure condition, and to have the technology and means to manage and predict the progression of this fault condition to component failure
Baruah and Chinnam (2005)	Prognostics builds upon the diagnostic assessment and are defined as the capability to predict the progression of this fault condition to component failure and estimate the remaining useful life (RUL)
Hess, Calvello and Frith (2005)	Predictive diagnostics, which includes determining the remaining life or period of proper operation of a component
Katipamula and Brambley (2005)	Address the use of automated methods to detect and diagnose degradation of physical system performance, anticipate future failures, and project the remaining life of physical systems in an acceptable operating state before faults or unacceptable degradations of performance occur
Wu, Hu and Zhang (2007)	The prediction of future health states and failure modes based on current health assessment, historical trends, and projected usage loads on the equipment and/or process
Heng et al. (2009)	The forecast of an asset's remaining operational life, future condition, or risk to completion
International Organization for Standardization (2015)	An estimation of time to failure and risk for one or more existing and future failure modes

2.2 Prognostic Framework

The prognostic framework systematizes a series of steps demanded to get a prognostic result, being a way to track and represent the information flow from the raw data until the result visualization of an appropriate remaining life estimation. Although standardization would be desired for purposes of performance evaluation, uncertainty quantification, and comparison between different models, there is no unified structure to guide such activity, only recommendations given in ISO 13374-1 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2003) and ISO 13381-1 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015). These standards attempted to generalize a conceptual framework aiming to provide basic CBM and PHM modules from data acquisition and analysis to health assessment, prognostic assessment, and advisory generation.

From different publications, it is possible to infer that there are similarities between the modules that orient their methodologies, even though sensible divergences could be noticed according to the technique chosen. The main modules presented in these architectures have been listed by Vachtsevanos et al. (2007) as:

1. A sensors collection and suitable monitoring strategies to collect and process data of critical process;

2. Failure modes, effects and criticality analysis module that determines and prioritizes according to the frequency of occurrence and their severity possible failure modes and catalogs systematically effect–root-cause relationships;

3. An operating-mode identification routine that determines the current operational status of the system and correlates fault-mode characteristics with operating conditions;

4. A feature extractor that selects and extracts from raw data features or condition indicators to be used by the diagnostic module;

5. A diagnostic module that assesses through online measurements the current state of critical machine components;

6. A prognostic module that estimates the remaining useful life of a failing component or subsystem;

7. A maintenance scheduler module, whose function is to schedule maintenance operations without adversely affecting the overall system functionalities, of which the machine in question is only one of the constituent elements.

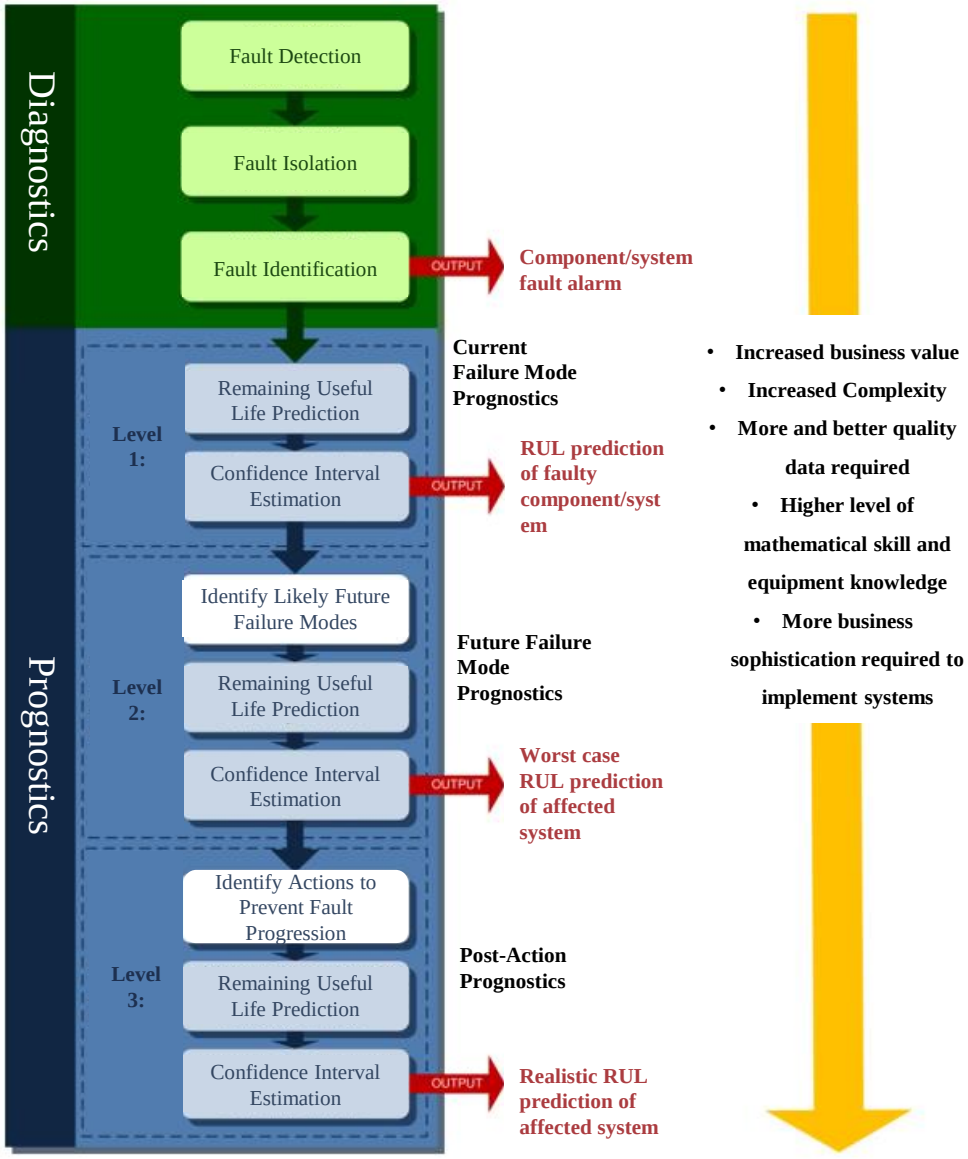
Other researchers (LEI et al., 2018), prefer explicitly declaring the health index (HI) construction a step of the prognostic scheme and discerning the health stage (HS) of the system by the indicator. They have categorized the HI types into two categories: physics HI and virtual HI. Physics HI is related to the physics of failure and is obtained from monitoring signals by statistical methods or signal processing techniques, such as the root mean square (RMS) of the vibration signal. And virtual HI results from the fusion of different information sources through a transformation that dissociates them from their physical meaning but displays a possible representation of the degradation trend. An example of the last one is the principal component analysis (PCA) applied to a set of sensor measurements (MELANI et al., 2021).

The health stage division in the Lei et al. (2018) framework shares similarity with the fault detection and diagnostic actions, but aims the particular goal of splitting a degradation pattern into different ¨health stages¨ according to variations in its characteristics. The HI evolution could show a gradual degradation, as well as exhibit a two-stage division – healthy and unhealthy - or a multistage one if there are discontinuities in the rate of change of the index.

In terms of prognosis and diagnosis integration, some authors diverge about considering them as parts of a unique framework, which is the case of Vachtsevanos et al. (2007), in contrast to that who prefer to treat them separately. In the current study, the first option is embraced and it is based on the process flow proposed by Sikorska, Hodkiewicz and Ma (2011), show in Figure 2, and Lei et al. (2018).

The diagnostic module is composed of three procedures as mentioned in Figure 2 and, according to Vachtsevanos et al. (2007), aims to detect, isolate and identify an impending or incipient failure condition, that is, when the affected element is still operational even though at a degraded mode. According to the same author, the goal of a diagnosis outcome in terms of performance is to determine the fault accurately, minimizing false positives and false negatives while reducing the time delay between the initiation and detection isolation of a fault event.

Figure 2: Flowchart illustrating the Sikorska, Hodkiewicz and Ma (2011) framework, based in ISO 13381-1.



Source: Sikorska, Hodkiewicz and Ma (2011)

The prognostic module placed in sequence receives as input the diagnostics data and returns the prognostic result as an output. The ISO 13381-1 divides this process into four actions, or one preprocessing and three prognostics types or levels (SIKORSKA; HODKIEWICZ; MA, 2011):

1. Data preprocessing: Prepares diagnosis information and compares it with declared failure symptoms definitions, identifying potential future failure modes and selecting a suitable prognostic model.

2. Existing failure mode prognosis (Level 1): Focuses on identifying the estimated time to failure (ETTF) of all incipient failures, equating the component's or system's RUL with the failure mode having the lowest ETTF, and iteratively repeat the process until the desired confidence in RUL is reached.

3. Future failure mode prognosis (Level 2): Sticks to assessing which are the most likely future failure modes and repeating the same process undertaken for existing failure modes for each of these potential failures to calculate an RUL with appropriate confidence, for potential future failure modes.

4. Post-action prognosis (Level 3): Identifies potential actions that could retard, halt or eliminate the progression of critical failure modes and prevent future ones, and then repeating the previous modeling processes with this information.

Lei et al. (2018) affirm that the usability and reliability of the business increase with the progression of the levels, but with an attached complexity cost, which manifests itself in the data requirement, supporting IT infrastructure, skilled personnel, and modeling depth. They also complements that the levels are interconnected, so each one demands a precedent outcome.

In other words, Level 1 urges to provide estimates for RUL of a component or system based on the evolution of the diagnosed faults, having each failure mode its prognostic model. In sequence, Level 2 calculates the probable effects of the identified failure modes on other potential failure modes and models the likely progression of the individual potential degradation paths to estimate the system health taking into account the critical scenario. Therefore, the step requires updated models for the subsequent faults and the modeling of the failure modes interactions. Level 3, in turn, incorporates the action plan formulated given the estimation and the maintenance actions influence over it, then updating again the prediction.

In the next section, various types of prognostic models will be described and stratified into a classification arrangement. Since a prognostic model is responsible for predicting the remaining life, it is likely to be a core component of the framework. Therefore, before delving into the details of deep learning, it is essential to have a brief overview of other techniques for the sake of comparison. This approach will help to understand the scope of application of deep learning and how it fits within the broader context of prognostic modelling.

2.2.1 Model Classification

Different alternatives to classify prognostic models have been presented in the literature with low consensus in the field about the most appropriate grouping schemes to classify remaining life models. Some proposed categorizations are disposed in (VACHTSEVANOS et al., 2007; JARDINE; LIN; BANJEVIC, 2006; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015; HENG et al., 2009; LUO et al., 2003; LEE et al., 2014), where, although similar, the class terminology cannot be considered standardized among the authors, due to the lack of explicit definition.

After examining the points raised above and the previous classification ideas, Sikorska, Hodkiewicz and Ma (2011) formulated a modified classification hierarchy, that allocates prognostics models inside four main groups as follows:

1. Knowledge-based models: Compare the similarity between an observed situation and a database of previously defined failures and deduce the life expectancy from previous events.

2. Life expectancy models: Determine the life expectancy of individual machine components considering the expected risk of deterioration under known operating conditions.

3. Artificial neural networks: Compute an estimated output for the remaining useful life of a component or machine, directly or indirectly, using a mathematical representation of the component or system that has been derived from observation data rather than a physical understanding of the failure processes.

4. Physical models: Calculate an estimated output for the remaining useful life of a component or machine using a mathematical representation of the physical behavior of the degradation processes. Types of physical models tend to be applied in specific failure modes and therefore will not be classified further.

It is fair to say that the conceptualization of an unified and immutable classification scheme will not be possible, because of the obsolescence of part of the techniques and the emergence of others, making the grouping criteria to be continually updated. Additionally, hybridization persists in the literature, as authors choose to combine models to increase performance, which makes the task of allocating such models in strict classifications rather difficult. Despite the issues, classification is essential to generalize characteristics in order to evaluate where and why a given model is the most viable for an application condition

(CA). Moreover, it contextualizes the current state of development in the field, during the taxonomy of novel approaches and the discovery of its qualities.

2.3 Deep Learning in Prognostics and Health Management

Deep learning embraces neural network learning models that have multiple layers of computational units, capable of decomposing higher-level abstract features in terms of other, more straightforward representations (GOODFELLOW YOSHUA BENGIO, 2016). As an extension of the single hidden layer networks, it is also suitable for supervised, unsupervised, and semi-supervised types of learning.

It emerges in PHM as a resource to solve previously intractable problems, improve performance over traditional techniques and reduce the effort to deploy prognostic systems (FINK et al., 2020) due to its advantages. Fink et al. (2020) listed some of them as the ability to automate the processing of a great amount of condition monitoring data, extract useful features from high dimensional heterogeneous data sources, learn functional and temporal relationships between and within the signal time series and transfer knowledge between different operating conditions and different units. Furthermore, DL contributes to attenuating the need for feature engineering in datasets composed of many monitored variables, which is demanding, by incorporating it inside the own network (FINK et al., 2020).

As stated by Fink et al. (2020), the current use of DL in PHM is mainly driven by developments in other domains, such as computer vision, and audio and natural language processing (NLP). These authors also emphasize that the growth of DL application in PHM doesn't imply concrete transfers to the industrial level.

Some DL PHM applications using sensors condition monitoring data were reviewed in order to support the present study. It could be observed that most reviewed authors chose to use more traditional deep neural networks (DNN) architectures, focusing on a few types of layers, rather than concentrate on more complex predefined models or design and optimize original very deep architectures. The employed networks encompass, feedforward, recurrent structures by means of long short-term memory (LSTM) and gated recurrent units (GRU), and convolutional structures with 1D and 2D CNN. The unidimensional receives time series data as input, and the bi-dimensional receives image encoding of time series or time-frequency analysis representation. Moreover, unsupervised time signal re-

construction techniques, also designated as residual-based approaches, which are usually done in the data-driven context by autoencoders, restricted Boltzmann machine (RBM), and its variations, deep belief network (DBN) and deep Boltzmann machine (DBM), have been recently explored.

The next subsections provide an panorama of the most conventional types of autoencoder architectures and the three traditional types of layers: convolutional, multilayer perceptron (MLP), and recurrent. The text also outlines some combined approaches that share a similar purpose with this study. This research was guided by recent literature reviews conducted by other authors in the field, and it has been carried out using the main academic search engines, such as Google Scholar, Mendeley, and Microsoft Academic.

2.3.1 Autoencoders

As a signal reconstruction technique, this architecture type can learn the normal system behavior and distinguish it from system states that differ from those observed under the labeled condition (FINK et al., 2020). Hence, not only they can be applied to perform anomaly detection, but also serve as support for diagnosis, if coupled to a classifier, or prognosis, whether the residue is extrapolated or fed into supervised models that map the remaining life. In both cases, compression capability stands out for enabling information fusion of multiple channels together with dimensionality reduction.

Ahmed, Wong and Nandi (2018) proposed an unsupervised feature learning algorithm using stacked autoencoders to learn feature representations from compressed measurements. The experimental results demonstrate higher accuracy levels than the existing techniques, even with extremely compressed measurements.

Tao, Liu and Yang (2016) studied different structures of a two-layer network designed by varying the hidden layer size and evaluated its impact on fault diagnosis. It should be noted that the input dimensionality in this publication was over one hundred; which indicated concerns about the large computational requirements and potentially overfitting problems due to so many parameters.

Some authors prefer input extracted features in the autoencoder (AE) to avoid entries with very large dimensions. In the denoising version of the AE, input is corrupted by noise before entering in the AE, but placed to reconstruct the original noncorrupted input.

Thirukovalluru et al. (2016) have used them for fault diagnosis and Lu et al. (2017) presented a detailed empirical study of stacked denoising autoencoders with three hidden

layers for fault diagnosis. This autoencoder variant uses multiple corruption or noise levels within all its layers and the network is trained to reconstruct a clean repaired version of the input – making it more robust.

Yan and Yu (2015) have applied it for anomaly detection. These authors acknowledge that representation learning can be a powerful tool for health management applications and demonstrated its potential by comparing handcrafted and deep learned features.

In the variational version of the AE, the input data is sampled from a parameterized prior distribution and the encoder and decoder are trained jointly such that the output minimizes a reconstruction error in the sense of the Kullback–Leibler divergence between the parametric posterior and the true posterior. In other words, the VAE receives a prior sample. The encoder compresses it into the latent space, then the decoder gets an entry sampled from the latent space and produces an output as close as possible to the encoder input.

Yoon et al. (2017) used autoencoders for RUL estimation and advocate that it is an effective architecture for dealing with the problem of insufficient labels in future reliability prediction

2.3.2 Convolutional Neural Networks

Convolutional networks (CNN) promote regularization by taking advantage of the sparse interactions, which attenuates the memory requirements and improves its statistical efficiency. Because of that, this network decomposes the input in simpler representations distributed over the filters around the layers. When combined with another useful property, called equivariance to translation, which could be translation invariant with the insertion of pooling, makes the network automates feature extraction, thus interesting for PHM purposes.

Janssens et al. (2016) made use of CNNs for rotating machinery condition monitoring. The input to the network was the result of a discrete Fourier Transform of the two accelerometers.

Babu, Zhao and Li (2016) built a CNN to predict the RUL of a system. The input was time series data from sensors. Since RUL is a continuous value, a regression layer was added. The authors were able to conduct a series of experiments and demonstrated how a CNN-based regression model can be used to outperform three other regression methods, that is, the multilayer perceptron, the support vector regression, and the relevance vector

regression.

2.3.3 Recurrent Structures

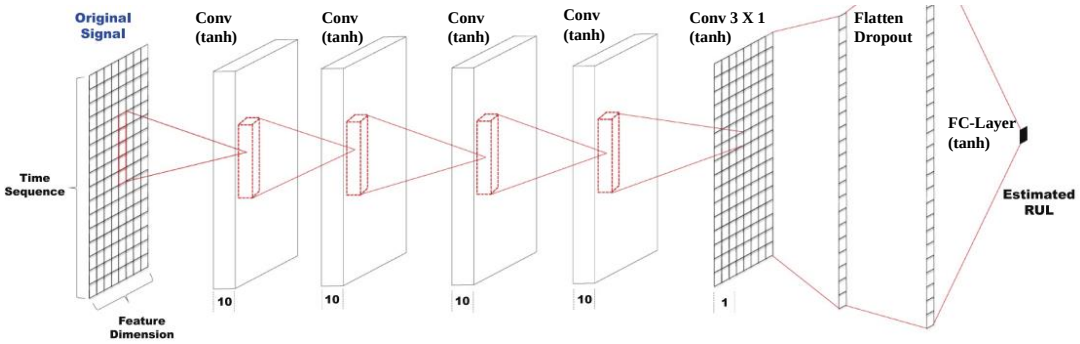
Networks specialized in dealing with sequential data are attractive for PHM purposes due to their capacity to discover and learn temporally intrinsic patterns in sensor readings.

Among the authors that explored the signal reconstruction approach are Malhotra et al. (2016), which combined LSTM layers in an encoder-decoder to get an unsupervised health index for a system using multi-sensor time-series data. The study concludes that LSTM-ED constructed HI learned in an unsupervised manner can capture the degradation in a system and that this HI can be used to learn a model for RUL estimation with equivalent performance to domain knowledge, or exponential and linear degradation models assumptions.

2.3.4 Combined approaches and others

Li, Ding and Sun (2018) used bidimensional deep convolutional neural networks with a time window applied to sample preparation and inputs direct raw normalized data – limited to the range [-1,1] - inside the DL model. Input width has been the time window length and depth equal to the number of features or channels. A piece-wise linear degradation representation has been used to express the health state progression until t_{eol}, that is, there has been made the implicit consideration that the unit works normally up to a point when it starts to degrade linearly. The validation has been carried out in the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset for aero-engine prognostics. The tested architecture is displayed in Figure 3.

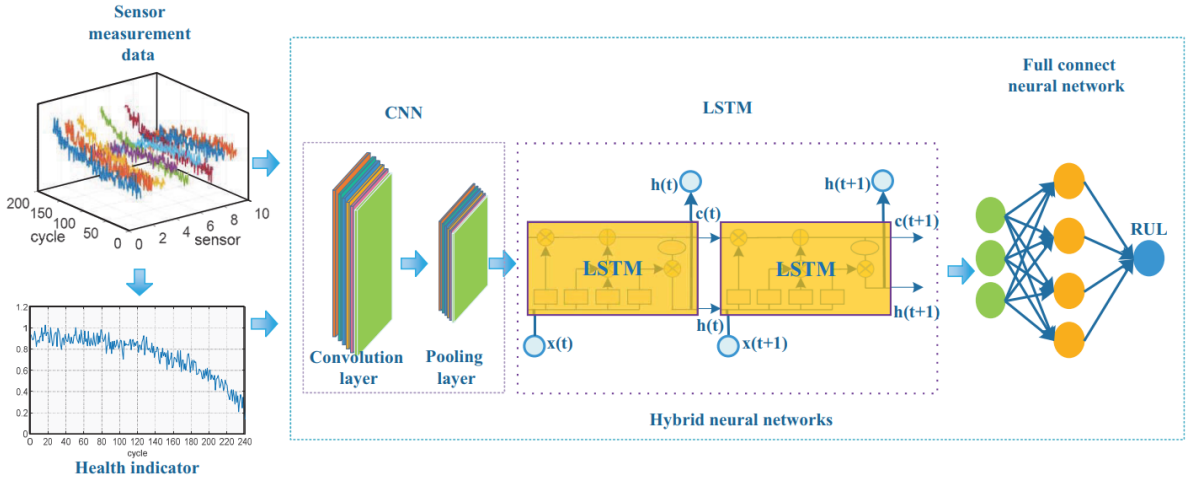
Figure 3: Deep convolutional combined with fully connected layers for remaining life regression elaborated by Li, Ding and Sun (2018).



Source: Li, Ding and Sun (2018)

Kong et al. (2019) also have worked in the same CMAPSS dataset. Its scheme is based on a hybrid deep neural network that combines CNN for spatial feature extraction with LSTM for temporal feature extraction and maps the input with a piece-wise function for the RUL. The input consists of preprocessed raw sensor data with redundant features removed together with a univariate health index build using polynomial regression in the preselected sensor data. Details of the architecture are shown in the Figure 4.

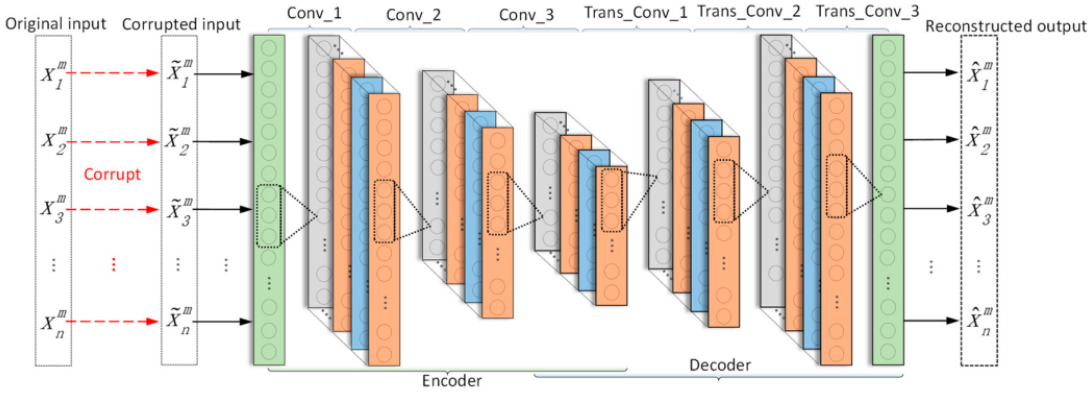
Figure 4: Hybrid neural network which comprises three kind of layers for RUL estimation elaborated by Kong et al. (2019)



Source: Kong et al. (2019)

Liu et al. (2019) create a method that combines 1-D convolutional networks with denoising convolutional autoencoder (1D-DCAE) for diagnosis – Figure 5. The authors intended to attenuate performance drop in practical applications, which usually have environmental noise. As expected in a DAE the original input is mixed with artificial noise and is a single normalized channel originating from a monoaxial accelerometer employed in both a bearing and a gearbox diagnosis experiment. In the first stage, the samples go through the DCAE and thus the denoised output feeds a deep CNN network in order to perform multilabel classification between fault types and normal conditions. According to the authors, the result indicated that the method achieves high diagnosis accuracy under low SNR conditions.

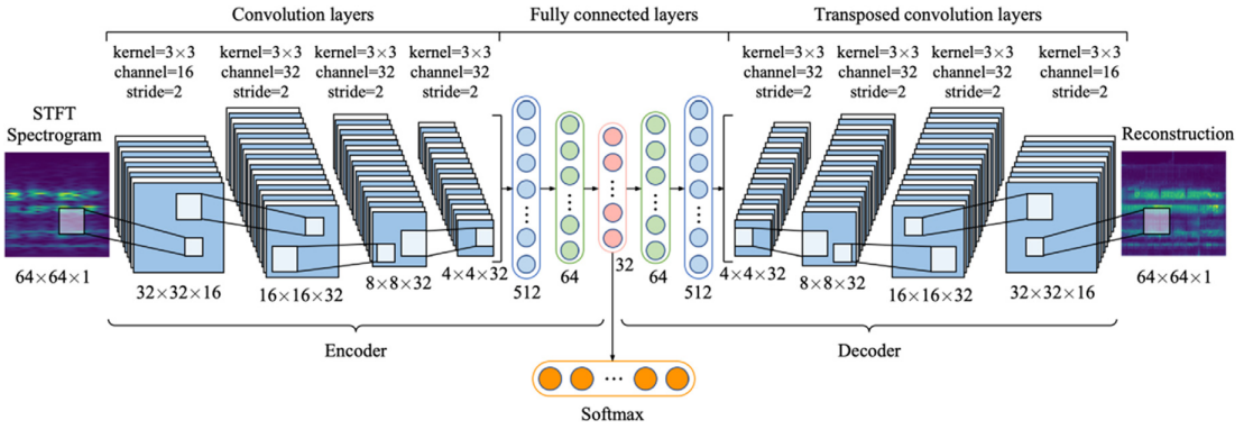
Figure 5: Denoising autoencoder with 1-D convolutional layers elaborated by Liu et al. (2019)



Source: Liu et al. (2019)

Wu et al. (2021) have contributed with a semi-supervised diagnosis architecture called hybrid classification autoencoder, which uses a softmax layer over the encoded features of the autoencoder for multilabel classification. In its approach, vibration data is pre-processed in a bi-dimensional entry by a short-time Fourier transform (STFT) and subjected to consecutive convolutional layers, as illustrated in Figure 6. Experimental validation has been performed in a publicly available dataset of moto-bearing signals. The authors also have realized a practical application in a hydro generator rotor diagnosed with a rub-impact fault between the turbine shaft and turbine guide bearing.

Figure 6: Hybrid autoencoder with 2-D convolutional and fully connected layers elaborated by Wu et al. (2021), whose the output of the latent space is subjected to a softmax activation function for multi label classification.



Source: Wu et al. (2021)

3 PROPOSED METHOD

This chapter presents a semi-supervised method for prognosis of remaining useful life that employ autoencoders. It also describes the main steps of this framework, including a algorithm for prognosis. At the end of the chapter, more details are given about the performance assessment and list of metrics is raised, some of them of authorial formulation.

3.1 General Description

The proposed approach relies upon generating a prognosis horizon for fault degradation patterns using the reconstruction error extrapolation of a deep autoencoder trained only with the machine's normal operating condition monitoring data. Instead of establishing a mapping between the observed set of signals and the evolution of the degradation and converting it to a health index, this work takes a different path. It focuses on detecting and isolating possible divergences in the monitoring measurements, which may indicate a fault, tracking their growth and extrapolating them to predict the machine's RUL. Such extrapolation is done using a set of more straightforward univariate functions with known behavior – one per channel – until a limit of divergence, signaling the failure's occurrence.

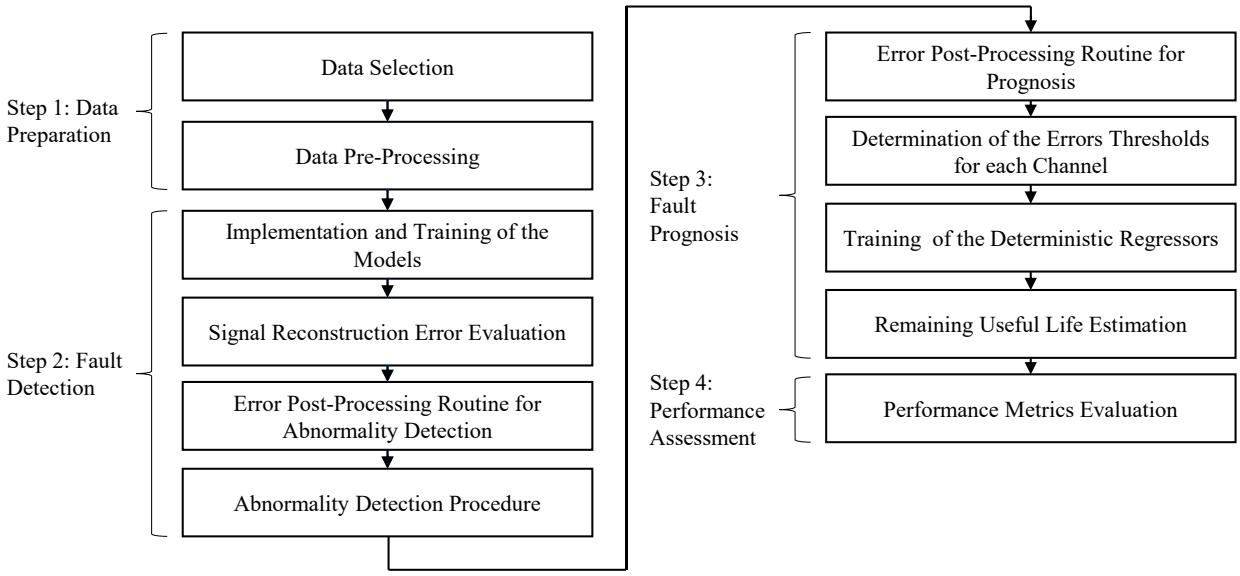
The reason for using this approach, in contrast with what is recently adopted in prognosis literature, is that there is a demand for models capable of following, recording, and interpreting machine behavior in the context of complex engineering systems (CES). Currently, industrial equipment is generally assisted with monitoring systems designed for operational intents whether by human or automated control. These systems are usually assisted with programmed fault alarms based on guidelines or empirical knowledge about the process, composing the resources for predictive maintenance. This type of setup is classified in Level 0 in terms of prognostic implementation readiness according to ISO 13381 classification (SIKORSKA; HODKIEWICZ; MA, 2011), i.e., they are CES with monitoring infrastructure capable of performing detection and sometimes fault identifica-

tion, but do not form a strong foundation for more sophisticated prognostic techniques, that demands intensive and systematical diagnostics capacity.

The proposed method constitutes a framework that has the potential to embrace all the past requirements - which are detection, isolation, and identification of the fault - for the remaining useful life prediction reaching Level 1 prognostics according to the same standard, agreeing with the above. In fact, since the autoencoder is a signal reconstructor and therefore can work as a hidden state reconstructor, diagnosis is possible because each channel could be compared individually to provide a multilabel classification of different kinds of faults.

Another motivation for adopting this approach is the unbalanced proportion of monitoring data between faulty and non-faulty conditions in an industrial scenario, since some failure events are rare in the operational history of certain kinds of equipment. This is due to slow deterioration evolution or early preventive intervention, which increases survivability. Hence, the availability of a great amount of data in the normal operating condition of a CES is attractive to the use of data-driven methodologies. The steps for implementing the prognosis framework and RUL prediction are expressed in the flowchart in Figure 7.

Figure 7: Flowchart describing the prognostic framework



Source: Author

3.1.1 Step 1: Data Preparation

The data selection comprises the procedure of selecting data in normal operational condition (NOC), being necessary beforehand to characterize this state with the help of a specialist or some reference criteria, for example, collecting data immediately after maintenance or an arbitrary time interval before the occurrence of a fault. It is interesting pointing out that there is no need to establish a perfect boundary in the transition of the conditions since it is desired to detect incremental abnormalities.

After that, the data is scaled using the given criteria, which could be done through a value range reference or by removing the mean and scaling to unit variance - standard score - according to the dataset profile. In the application example, standard scaler has been used, because its the most appropriate to avoid scaling distortions between variables and punctual variability induced by outliers.

It is worth emphasizing that only NOC-labeled data is applied to calibrate the scaler to avoid distortions in the set designated to train the networks. Following this, the data is reshaped into a set of subsequences of size n that will supply the models. These subsequences are generated through a moving window with a temporal iteration step of p, thus allowing overlapping of $n - p$ entries. Each sample has shape (n, m), being n the subsequence size and m the number of channels - sensors inputs.

For this work, NOC-labeled data is split between train and validation sets to prepare the neural network in a 9 to 1 proportion. The terminology test set will be designated exclusively to refer to data not applied in the DNN training and tunning process, including abnormal-labeled data. The validation samples are placed at the right end of the NOC-labeled interval, most recent in time, to ensure that there is no interference caused by overlapped samples and that the independence between the partitions is conserved.

3.1.2 Step 2: Fault Detection

The DNN autoencoder models are programmed according to the hyperparameters specifications in Table 2 and their structures are illustrated in Figure 8. Three different kinds of layers will be used: MLP, LSTM, and 1D-Convolutional (1D-CNN or Conv-1d), commonly used in the setup of DL models to monitor signals in the literature. Albeit MLP could express poor performance in comparison with the other layers (ZHAO et al., 2020), it is applied in this study as a reference to analyze the models. Thus, a minimum requirement for them is to outperform a classic MLP perceptron architecture. Recur-

rent neural networks, especially LSTM, are widely used for PHM applications. Moreover, Conv-1d is an alternative to apply convolutional operations into time-series data without demanding transformations to the bi-dimensional spatial space, which is time-consuming. Also, Conv-1d is less computationally expensive by having fewer parameters. The implemented models are tuned using a grid-search over a group of hyperparameters and the impact of these variables on the chosen performance metrics is discussed in Section 4.1.2.

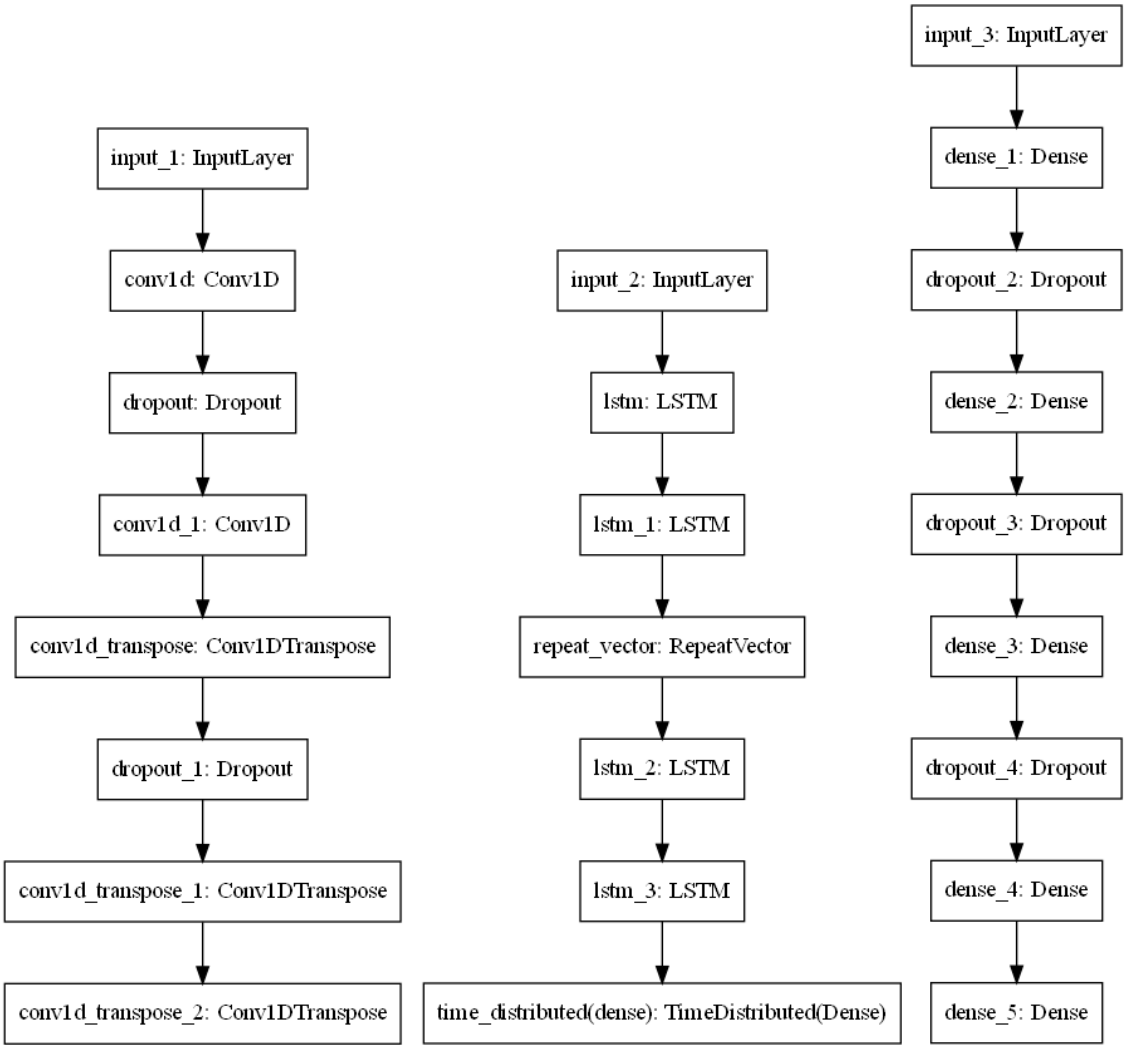
Rosa et al. (2022b) investigated the sensitivity of AE architectures hyperparameters over its abnormality detection performance. The study concluded that some specific hyperparameters influence the model outcomes more than others, therefore serving as a reference for the search space definition. Although easy to implement, grid-search is an exhaustive procedure, being inefficient without prior knowledge of the search space near the optimality. Some alternatives are the random search or search based on Bayesian optimization theory (BERGSTRA et al., 2011).

Table 2: Hyperparameters specification of the analyzed autoencoders in Section 4.1.1.

Type	Hyperparameters	Definition
Global	Number of layers - Encoder only	2
	Dropout rate*	0,3
	Loss Function	MSE
	Optimization Technique	Adam
	Subsequences Size	200,
	Validation data fraction	10%
1D-CNN	Strides	2
	Learning Rate	0,001
	Number of filter units	(32,16)
	Kernel Size	(15,10)
	Padding	'same'
	Activation Function	LeakyReLU
	Epochs	70
LSTM	Activation Function	tanh
	Number of LSTM units	(64,32)
	Epochs	70
MLP	Activation Function	LeakyReLU
	Neurons	(32,16)
	Epochs	100

The reconstructed signal subsequences outputted from the trained AE models are compared with the actual signal observations, and the reconstruction error (RE) is evaluated. RE is computed as a mean squared error (MSE) function applied in a subsequence for each channel, so it is possible to inspect discrepancies individually. The subsequences

Figure 8: Representation of the autoencoders architectures. Each box indicates a layer, and the arrows indicate the information flow.



(a) 1d-Convolutional (b) LSTM (c) MLP

Source: Author

are addressed by the time index of the last observation t_i; then, the RE is also assigned for this position. Thereby, the reconstruction error follows the notation below:

$$RE_{i,j} = \frac{\sqrt{\|s_{i,j} - s^r_{i,j}\|}}{n} \tag{3.1}$$

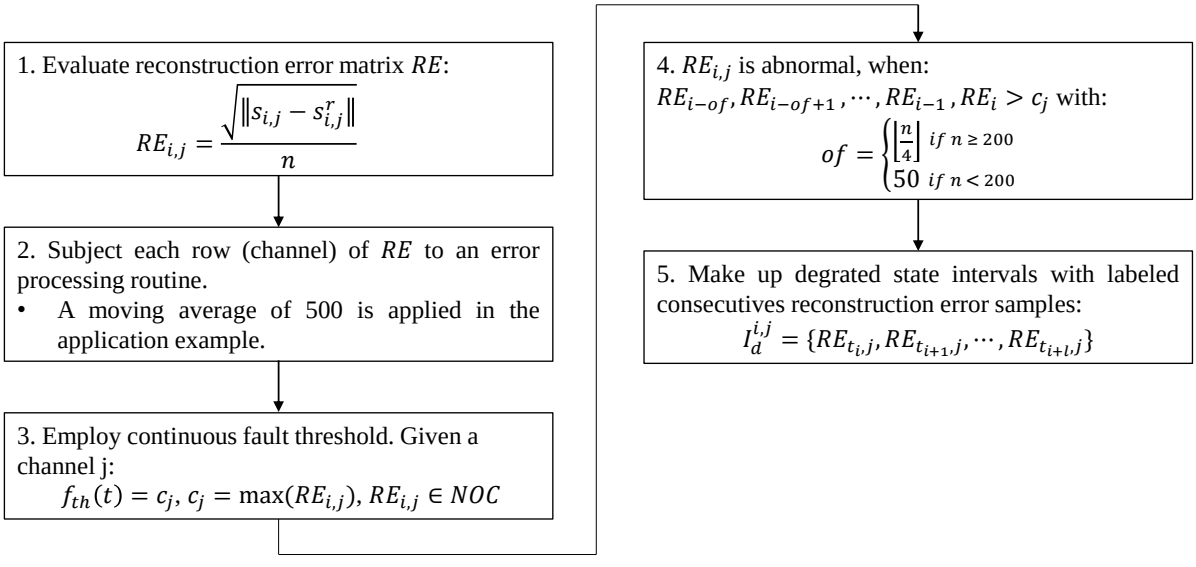
in which $RE_{i,j}$ is the reconstruction error for the subsequence $s_{i,j} = \{s^{(0)}_{i-n,j}, ..., s^{(n)}_{i,j}\}$ with temporal index i, channel index j and size n. $s^r_{i,j}$ corresponds to the reconstructed subsequence mimicking $s_{i,j}$.

The abnormality detection procedure comes afterwards, employing the reconstruction error matrix RE, whose entries are defined by Equation 3.1, to build a set of error

threshold functions f_{th} that is used to classify whether or not a data entry is abnormal. First, it is important to note that the $RE_{1:n,j}$ series are subject to local variability due to the outliers that could come from the sensor's readings. For example, in machines with more than one operational mode or those with intermittent operation, the working routine is cyclical, having unstable behaviors during state transitions or due to variations in the cycle periods. Examples are the take-off and landing of aircraft or the switch between generation and motorization modes in hydro-generators. Moreover, it could not be a trivial task to characterize state transitions, even for experts in the process, being a part of the data selection step of this study. This fluctuation could severely affect the method's abnormality detection capacity and must be considered in the definition of $f_{th}(t)$ and interpretation of the entries of RE.

The main part of Step 2 is summarized by the flowchart presented in Figure 9 and exemplified by the graphics in Figure 10. The non-conformities are detected in this work by using a set of continuous threshold functions $f_{th}(t) = c_j$, being c_j the maximum value between the post-processed RE samples labeled as NOC for the j^{th} channel. Samples of the post-processed $RE_{1:n,j}$ that exceed c_j are labeled as abnormalities. Sometimes only the post-processing of RE is not enough to avoid the occurrence of false positives, which are provoked by point-wise or small cluster points addressing. To highlight the cumulative abnormality resulting from the monotonic growth of the degradation pattern, an offset of consecutive abnormal labeled points is taken as a requirement for pointing out the beginning of the degradation.

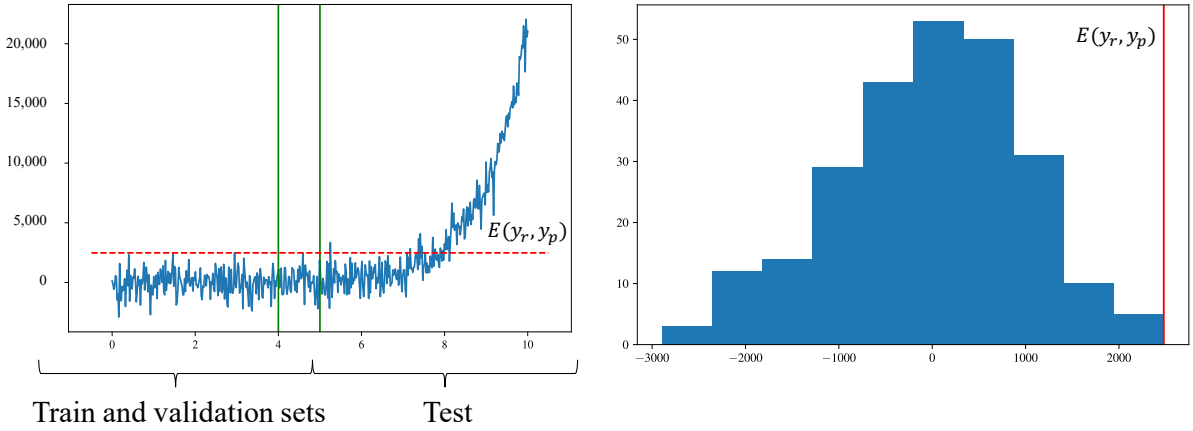
Figure 9: Flowchart systematizing the abnormality detection procedure.



Source: Author

In 10a, there is a temporal progression of the reconstruction errors for an arbitrary variable. Train and validation sets correspond to NOC, while the test set is in the degraded condition. $E(y_r, y_p)$ is the maximum error between the samples in the condition labeled as normal and it is equivalent to $max(RE_{ij})$ in the Figure 9 flowchart. The localization of E in the samples' distribution is displayed in 10b. As the monotonic pattern evolves, it exceeds E, and if a certain quantity of consecutive RE, called of, keeps above the limit, the abnormality is registered.

Figure 10: Exemplification of the procedure in Step 2.



(a) Temporal progression

(b) Distribution

Source: Author

3.1.3 Step 3: Fault Prognosis

The remaining useful life estimation is developed from the samples inside the degraded state intervals (I_d) provided by the abnormality detection procedure. I_d is defined as a set of consecutive reconstruction error samples, meaning that:

$$I_d^{i,j} = \{RE_{t_i,j}, RE_{t_{i+1},j}, \cdots, RE_{t_{i+\ell},j}\} \tag{3.2}$$

where $I_d^{(i,j)}$ is the i^{th} interval with cardinality ℓ+1 for the channel j, where $\{t_i, t_{i+1}, \cdots, t_{i+\ell}\}$ is the ordered set of temporal indexes addressed for the RE's. An additional notation is $I_d^{t_d,i,j}$, which emphasizes the temporal placement of I_d, which is agreed to be the time of the first abnormal sample within I_d. For convenience, the positional argument could be suppressed occasionally, so: $I_d^{i,j} \equiv I_d^{t_d,i,j} \equiv I_d^{t_d,j}$. Also, $\mathbf{I_d}^{(j)}$ represents the superset of all I_d's in the channel j, then $\mathbf{I_d}^{(j)} \supset I_d^{(i,j^*)} \forall i$, such that $j^* = j$. And to symbolize the union of sets more conveniently, the notation $\mathfrak{I}_d$ is employed, such that $\mathfrak{I}_d = \cup_{j=0}^{m} \cup_{i=1}^{n} I_d^{(i,j)}$ and $\mathfrak{I}_d^{(j)} = \cup_{i=1}^{n} I_d^{(i,j)}$.

These samples could either be subjected to another post-processing routine specially designed for prognosis or the same routine already made for abnormality detection. Postulating it, the initial goal for RUL evaluation is to determine the prognostic error threshold for each channel, which is equivalent to the failure limit of a built health index or measured quantity. As the RE cannot directly relate to future variations on the input channels - unless explainability techniques are coupled to the DNN - it is necessary to take past failures events as references to determine those thresholds. Thus, the prognostic error threshold $PE_{th}^{(j)}$ for the j^{th} channel is given as an average of k reconstructed error samples and n fault observations before the failure.

The next step is iteratively fitting curves and executing extrapolations from the first detected abnormality until the error threshold for each channel to get the RUL prediction at the time t. At the instant t, there can be more than one estimation because the degradation evolution of each channel is treated independently and fitted to an univariate function. Therefore, a decision criterion is demanded to provide a singularized prediction, which is done by observing curve fitting metrics, prognostic threshold variability, and the monotonicity of the generated profiles together with the values of the produced estimations.

The pseudocode below systematizes details about the aforementioned step:

Algorithm 1 Estimation of the RUL for a experimental fault event with the made assumption that the real remaining life is known for study purposes.

Input: $\mathbf{I_d}$, PE_{th}, f, t_{eol}
Output: r^*

$t_{init} \leftarrow min(t_d(I_d^{(0,j)}) : 0 \leq j \leq m)$
for $t_i = t_{init}$ to t_{eol} **do**
  for $j = 0$ to m **do**
    if $\exists I_d^{(t_{ds}:t_{df},j)}$ such that $RE^{(t_i,j)} \in I_d^{(t_{ds}:t_{df},j)}$ **and** $t_{ds} \leq t_i \leq t_{df}$ **then**
      $\mathbf{x} \leftarrow t_{ds} : t_i$
      $\mathbf{y} \leftarrow I_d^{(t_{ds}:t_i,j)}$
      for $i = 0$ to $n(\mathbf{f})$ **do**
        $c^{(i)} \leftarrow$ Solve: Least squares to $f^{(i)}$ in $(\mathbf{x}, \mathbf{y})$
        $t_{eol}^* \leftarrow$ Solve: $f^{(i)}(c^{(i)}, \mathbf{x}) - PE_{th}^{(j)} = 0$
        $r^{*(i,j)} \leftarrow t_{eol}^* - t_i$
      end for
      $i^* \leftarrow D_{f1}(R^2(f^{(i)}(c^{(i)}, \mathbf{x}), \mathbf{y}) : 0 \leq i \leq n(f))$
      $r^{*(j)} \leftarrow r^{*(i^*,j)}$
      $Mon^{(j)} \leftarrow Mon(f^{(i^*)}(c^{(i^*)}, \mathbf{x}), \mathbf{y})$
      $R^{(j)} \leftarrow R^2(f^{(i^*)}(c^{(i^*)}, \mathbf{x}), \mathbf{y})$
    end if
  end for
  $r^{*(t_i)} \leftarrow D_{f2}(\langle Mon^{(j)}, R^{(j)}, \sigma(PE_{th}^{(j)}), r^{*(j)} \rangle : 0 \leq j \leq m)$
end for

The Algorithm 1 has three main loops that permeate the prognosis procedure in a given inspection interval when an abnormality is detected. The loops, from the most to the least nested, iterate respectively through curves (Loop 1), channels (Loop 2), and in time (Loop 3). The first one adjusts the function shape for a channel m at time t_i using the least squares method and estimates the t_{eol} and thus the RUL. The second one, in turn, evaluates RUL and curve-fitting metrics for each one of the channels with degradation labeled in t_i using the decision function Df_1. And by the end, the third loop decides whether a prediction is made at the time t_i and its value is deliberated by the decision function Df_2. Decisions functions are subroutines that hierarchically dispose of the most likely remaining life prediction(s), after inputting a list of them, by means of the analysis of a set of curve-fitting metrics. Df_1 employs only R^2 in the sorting and eliminates nonsensical outcomes and those below an established fitting limit. The final result comes from the mean of the remnants' occurrences. Df_2 considers monotonicity besides R^2 and weights the last one in a fraction of 0.8 out of 1.

Furthermore, the curve fitting is executed using a non-linear least-squares problem with bounds on the variables. The objective is to find a local minimum of the cost function $F(x)$, that is:

$$\text{Minimize } F(\mathbf{c}) = \sum_{i=1}^{N} \rho\left(r_i(\mathbf{c})^2\right) \tag{3.3}$$

Subject to:

$$\mathbf{c} \geq 0$$

where $\mathbf{c}$ is a vector of estimable parameters, N is the number of available data points, $\rho(s)$ a scalar loss function that reduces the outliers' influence, and $r_i(\mathbf{c})$ is the component i of the vector of residuals $\mathbf{r}$. Residuals are understood as the difference between the prediction of a model function $f(x, c)$ and a set of data points $\{(x_i, y_i) \mid i = 1, \ldots, N\}$, so $r_i(\mathbf{c}) = f(x_i, \mathbf{c}) - y_i$.

Minimization is performed through the trust-region reflective algorithm implemented in an open-source scientific computing library. The used curves are disposed in Table 3 and represent common degradation patterns found in mechanical components (LEI et al., 2018; LIU; FAN, 2022).

The value of the function $f^{-1}(RE)$, inverse of $f(t)$, at the point $p = PE_{th}^{(j)}$ gives the component $t_{eol}^{*(j)}$. Thus, the estimated RUL at the instant t_i is $r(t_i) = t_{eol}^{*(j)} - t_i$ for the

Table 3: Selected model functions $f^{(i)}(c^{(i)}, \mathbf{x})$, that composes the vector f subject to be minimized in accordance with the Algorithm 1.

Function $f^{(i)}$	Behavior
1	$c_1 x + c_0$
2	$c_2 x^2 + c_1 x + c_0$
3	$c_1 log(x) + c_0$
4	$c_2 e^{c_1 x} + c_0$

fitted curve.

3.1.4 Step 4: Performance Assessment

Performance assessment is executed with dedicated performance metrics for comparing the autoencoders during the training process, abnormality detection and prognostic. The AE convergence is observed through the training and validation set loss on the last training epoch. Abnormality detection capacity is measured by detection coverage, d, and false-positive coverage, f, respectively:

$$d = \frac{n(\mathfrak{I}_d^r \cap \mathfrak{I}_d^*)}{n(\mathfrak{I}_d^r)} * 100 \tag{3.4}$$

$$f = \frac{n(\mathfrak{I}_n^r \cap \mathfrak{I}_d^*)}{n(\mathfrak{I}_n^r)} * 100 \tag{3.5}$$

The indicator d measures the ratio between samples correctly signaled by the method as abnormalities and the real set of degradation occurrences and f relates to the ratio of NOC samples highlighted on the same condition and the real entries in the normal state.

Other indicators used in evaluating performance are the discontinuity index $I_{dc}^{(j)} = (N_{int}^{(j)})^{-1}$, where $N_{int}^{(j)}$ is the number of intervals where abnormality was detected and the time interval between the first spotted abnormal point t_{sp} and the concrete tipping point for the degraded stage t_d: $\Delta_{sp} = t_{sp} - t_d$.

The prognostic capacity, in turn, is quantified by the root-mean-squared error ($RMSE$), NASA´s scoring function (s) (SAXENA et al., 2008; CHAO et al., 2022), adaptation (ns-score), and the prognostic horizon. The first two are respectively defined as:

$$s = \sum_{k=1}^{N^*} \exp\left(\beta \left|\Delta^{(k)}\right|\right) - 1 \tag{3.6}$$

$$RMSE = \sqrt{\frac{1}{N_*}\sum_{k=1}^{N_*}\left(\Delta^{(k)}\right)^2} \tag{3.7}$$

in which N^* indicates the total number of RUL estimations, $\Delta^{(k)}$ is the difference between the predicted and the real remaining life of the k^{th} sample, $\Delta^{(k)} = r(t_k) - r^*(t_k)$, and β is $\frac{1}{14}$ if RUL is underestimated, and $\frac{1}{10}$ otherwise. These values were intended to the PHM Conference Data Challenge 2008 Edition. The s metric is not symmetric and penalizes overestimation more than underestimation (CHAO et al., 2022).

The prognostic horizon is defined as the time interval between the time $t_{\top(C)}$ when a made prediction first meets specified performance criteria C that continues being satisfied until t_{eol} for all $t^{(i)}$ such that $t_{\top(C)} \leq t^{(i)} \leq t_{eol}$, thus:

$$H_{\top(C)} = t_{eol} - t_{\top(C)} \tag{3.8}$$

Moreover, some metrics proposed by Saxena et al. (2010b) may be used for auxiliary performance inspection, i.e., not designated for a specific finality of tunning or validation in this study within the models' comparison schema. These metrics are the α-λ performance, relative accuracy and cumulative relative accuracy.

The $\alpha - \lambda$ *accuracy* is a binary metric that evaluates whether the prediction falls in an α-bounded interval centered in the actual RUL value at a specific time instance t_λ in order to assess whether the following condition is met (SAXENA et al., 2010a):

$$(1-\alpha)\cdot r^*(t_\lambda) \leq r^t(t_\lambda) \leq (1+\alpha)\cdot r^*(t_\lambda) \tag{3.9}$$

in which α is the accuracy modifier, λ is a time window modifier such that $t_\lambda = t_p + \lambda(t_{EoL} - t_p)$ and $\lambda \in [0,1]$. For a probabilistic estimation of the remaining life Equation 3.9 is expressed as (SAXENA et al., 2010b):

$$\alpha - \lambda \text{ Accuracy } = \begin{cases} 1 & \text{if} \quad \pi\left[r\left(t_\lambda\right)\right]_{-\alpha}^{+\alpha} \geq \beta \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where β is the minimum acceptable probability mass, $r\left(t_\lambda\right)$ is the predicted RUL at time t_λ, and $\pi\left[r\left(t_\lambda\right)\right]_{-\alpha}^{+\alpha}$ is the probability mass of the prediction PDF within the α-bounds that are given by $\alpha^+ = (1+\alpha)\cdot r(t_\lambda)$ and $\alpha^- = (1-\alpha)\cdot r(t_\lambda)$.

Relative accuracy (RA), in turn, is defined as a error measurement in RUL prediction relative to the actual RUL $r^*(t_i)$ at a specified t_i, then:

$$RA_i^l = 1 - \frac{\left| r^l\left(t_i\right) - \left\langle r^{*l}\left(t_i\right)\right\rangle \right|}{r^l\left(t_i\right)} \tag{3.11}$$

where l is the index for the l^{th} prognostic experiment, $r^l\left(t_i\right)$ is the ground truth remaining life at time t_i, and $\left\langle r^*\left(t_i\right)\right\rangle$ is an appropriate central tendency point estimate of the predicted RUL distribution at time index t_i.

Since relative accuracy is expressed punctually, to get an overall view of the algorithm behavior over time it is necessary to aggregate the measurements as a normalized weighted sum of relative accuracies for all the predictions in one prognosis experiment, resulting in a metric called cumulative relative accuracy, which is:

$$CRA_i^l = \frac{1}{n(\mathbf{p}_i)} \sum_{i \in \mathbf{p}_i} w\left(r^l(t_i)\right) RA_i^l \tag{3.12}$$

where $w\left(r^l\right)$ is a weight factor as a function of RUL at all time instances, $\mathbf{p}_i$ is the ordered set all time indexes before t_i and $n(\mathbf{p}_i)$ is the cardinality of the set $\mathbf{p}_i$.

Besides the metrics based on accuracy, it is also important to mention the monotonicity criteria that is applied as an input of the decision function for the RUL discrimination at the time t_i, previously elucidated. Lei et al. (2018) has gathered some ways to evaluate it starting from the assumption that degradation of machinery is an irreversible process and then should be linked with monotonic increasing or decreasing trends.

There are monotonicity metrics based on the count of finite differences $d/dx = x_{k+1} - x_k$ of a health index sequence $X = \{x_k\}_{k=1:K}$ with x_k being the value of HI at time t_k (LEI et al., 2018; LIAO, 2014; ZHANG; ZHANG; XU, 2016; JAVED et al., 2015). One is described as:

$$\mathrm{Mon}_1(X) = \frac{1}{K-1} \mid \text{ No. of } d/dx > 0 - \text{ No. of } d/dx < 0 \mid \tag{3.13}$$

where K is the number of elements of the set X, $No.\,of\ d/dx > 0 - No.\,of\ d/dx < 0$ represents the number of positive and negative differences, respectively, and then $\mathrm{Mon}_1(X)$ quantifies the absolute difference between them, normalizing it for the interval $[0, 1]$. Other indicator presented by Lei et al. (2018) also comprises second order derivatives as follows:

$$\begin{cases} \mathrm{Mon}_{2+}(X) = \frac{\text{No. of } d/dx>0}{K-1} + \frac{\text{No. of } d^2/d^2x>0}{K-2} \\ \mathrm{Mon}_{2-}(X) = \frac{\text{No. of } d/dx<0}{K-1} + \frac{\text{No. of } d^2/d^2x<0}{K-2} \end{cases} \tag{3.14}$$

where the second order derivatives of X are denoted as d^2/d^2x. In that case, the positive monotonicity $\mathrm{Mon}_{2+}(X)$ and the negative one $\mathrm{Mon}_{2-}(X)$ are calculated individually.

Table 4 gives an overview about the advantages, disadvantages and application context of each discussed metric.

Table 4: Metrics summary.

Metric	Function	Advantages	Disadvantages	When to use
d	Gets the ratio between samples correctly signaled by the method as abnormalities and the real set of degradation occurrences.	Easy to interpret. It is bounded and normalized (0 – 100%). Strong argument of the model detection capacity, when combined with f.	Can not be used for punctual abnormalities. Does not take into account discontinuities.	Observation of the start and progression of monotonic growth patterns.
f	Measures the false-positive occurrences during the abnormality detection.	Easy to interpret. It is bounded and normalized (0 – 100%). Strong argument to declassify a detection model.	Can not be used for punctual abnormalities. Does not take into account discontinuities. Assumes very small values when only a few occurrences happen. It is influenced by the size of the time window observed	Observation of the start and progression of monotonic growth patterns. When the dimensions of the inspection intervals are conserved.
I_{dc}	Quantifies the density of discontinuities in detecting an abnormality caused by a monotonic growth pattern.	A simple way to provide information about the discontinuities in the inspected interval.	Tends to return very small values to large intervals, requiring reformatting the result, which in turn could complicate the interpretability.	When discontinuities could affect subsequent prognosis performance.
s	Computes the prediction error considering overestimation and underestimation effects.	Asymmetry: allows recognizes which models overestimate.	Parameterized to a very specific application. Generalization may not suit well other domains.	When penalizing RUL overestimation over underestimation is required.
Δ_{sp}	Measures the latency of the detection model, i.e., the time interval between the first spotted abnormal point tsp and the concrete tipping point for the degraded stage.	Easy to interpret. Could provide insights about the detection in different conditions when analyzed statistically.	It is not robust to punctual abnormalities. In practical applications, it is complicated to know the tipping point to abnormality precisely.	Validation of detection algorithms.
$RMSE$	Computes the prediction error	Traditional and easy to interpret. With complementary modifications (weigh factor, subinterval fractionalization), could dismiss the use of additional metrics.	It is not dimensionless. Can not track error progression.	Compare models with stable or stationary error progression for the long part of the inspection interval. Compare models used in different application domains or with distinct working principles.

t_{fpt}	Indicates the first prediction time.	Easy to interpret. Does not require knowing the tipping point of an abnormality.	Only informs when the prognosis has started, but does not provide any information besides that. Sensible to punctual outliers of the prediction error.	Know when the prognosis started and evaluate the percentual of the asset's life covered by the predictions.
$H_{\top(C)}$	Indicate the time when prediction first meets specified performance criteria that continues being satisfied until the t_{eol}, usually prediction error.	Easy to interpret. Appropriate eliminatory criteria for model selection.	It is not dimensionless. Sensible to punctual outliers of the prediction error.	Model selection or performance ranking.
$\alpha - \lambda$	Evaluates whether the prediction falls in an alpha-bounded interval centered in the actual RUL value at a specific time. Could be used to find the time, when the condition continues being satisfied until the t_{eol}.	Adaptable for probabilistic predictions of the RUL. Determines linear boundaries.	It is not easy to understand without a graphical representation. Requires parameterization.	When the demand for more accurate precisions increases with the asset's lifetime. Estimation is a random variable.
RA	Computes an accumulation of prediction errors relative to RUL. When the weight factor is one, is equivalent to the mean.	Dimensionless quantity. Normalized upper bounded quantity.	Can not track prediction error progression. Does not allow discerning between underestimation and overestimation of the RUL.	Compare models with unstable or transient error progression for a long part of the inspection interval.
CRA	Computes the prediction error relative to RUL.	Normalized upper bounded quantity.	Weight factor defined by the user. There is no prior rule to guide its choice.	When it is necessary evaluate the entire prediction error profile, which in turn is unstable or transient.
$Mon(X)$	Intend to quantify the monotonicity of a trend.	Does not require knowing the ground truth RUL.	Unstable in outputs not post-processed, then sensible to local noise and outliers.	Trend characterization is necessary for curve-fitting.

4 RESULTS AND DISCUSSIONS

This chapter examines the resourcefulness of the formulated method with an application example in a simulated database divided into two parts. First, it shows the outcome from the implementation of three models with fixed hyperparameters, each one with a fundamental type of layer, and one baseline model, to verify the effectiveness of the autoencoders. Thereafter, reports a grid-search experiment with a large collection of models to test the tuning capacity and perform sensitivity analysis through the hyperparameters modulation. At the end, it discusses the limitations, further improvements, and future opportunities.

4.1 Application example in CMAPSS Dataset

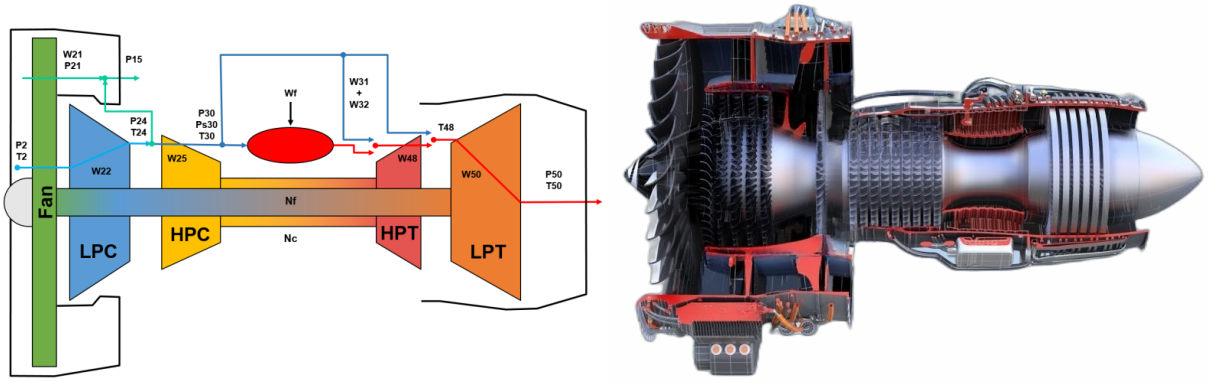
The database chosen for the study is a variant of the Commercial Modular Aero-Propulsion System Simulation (CMAPSS), publicly available and recognized as one of the datasets frequently used for benchmark prediction algorithms. It has recently been updated after a joint work between NASA and ETH Zurick's intelligent maintenance systems center so that the amount of sensor samples has been increased to 1 Hz, making it suitable for the study of the models oriented for large volumes of data.

The CMAPSS-2 (CHAO et al., 2021) is composed of a set of synthetic run-to-failure (RTF) trajectories, that is, with artificial degradation, of nine units of turbofan engines, produced by the simulator from the input of real flight conditions, which is characterized by the scenario descriptor variables: altitude, Mach number, throttle-resolver angle (TRA) and total inlet blade temperature. The base is divided into six units designated for training and another three for testing, with operating conditions slightly different from the others. In this study, only the training data from CMAPSS-2 were used, which does not compromise the feasibility study, since the tested model is semi-supervised and, therefore, uses only a part of the samples from each unit for training.

The inserted degradation pattern is of continuous type and is divided into four states:

degradation condition at the beginning of the operation; normal state; a transition zone between normal to abnormal condition; and an abnormal state. The simulation considers the alternating presence of failure modes in the main sub-components of the motor: fan, LPC, HPC, HPT, and LPT. Their deteriorations are modeled by adjustments in flow capacity and efficiency. More information about the modeling can be found in Chao et al. (2022). Figure 11 outlines the allocation of the main subsystems of a turbofan engine.

Figure 11: Representation of the main subsystems of the turbofan engine simulated in CMAPSS. From left to right: Fan, low pressure compressor (LPC), high pressure compressor (HPC), combustion chamber, high pressure turbine (HPT) and low pressure turbine (LPT).



(a) Schematics

Source: Chao et al. (2021)

(b) Cutaway view

Source: Chesky (2017)

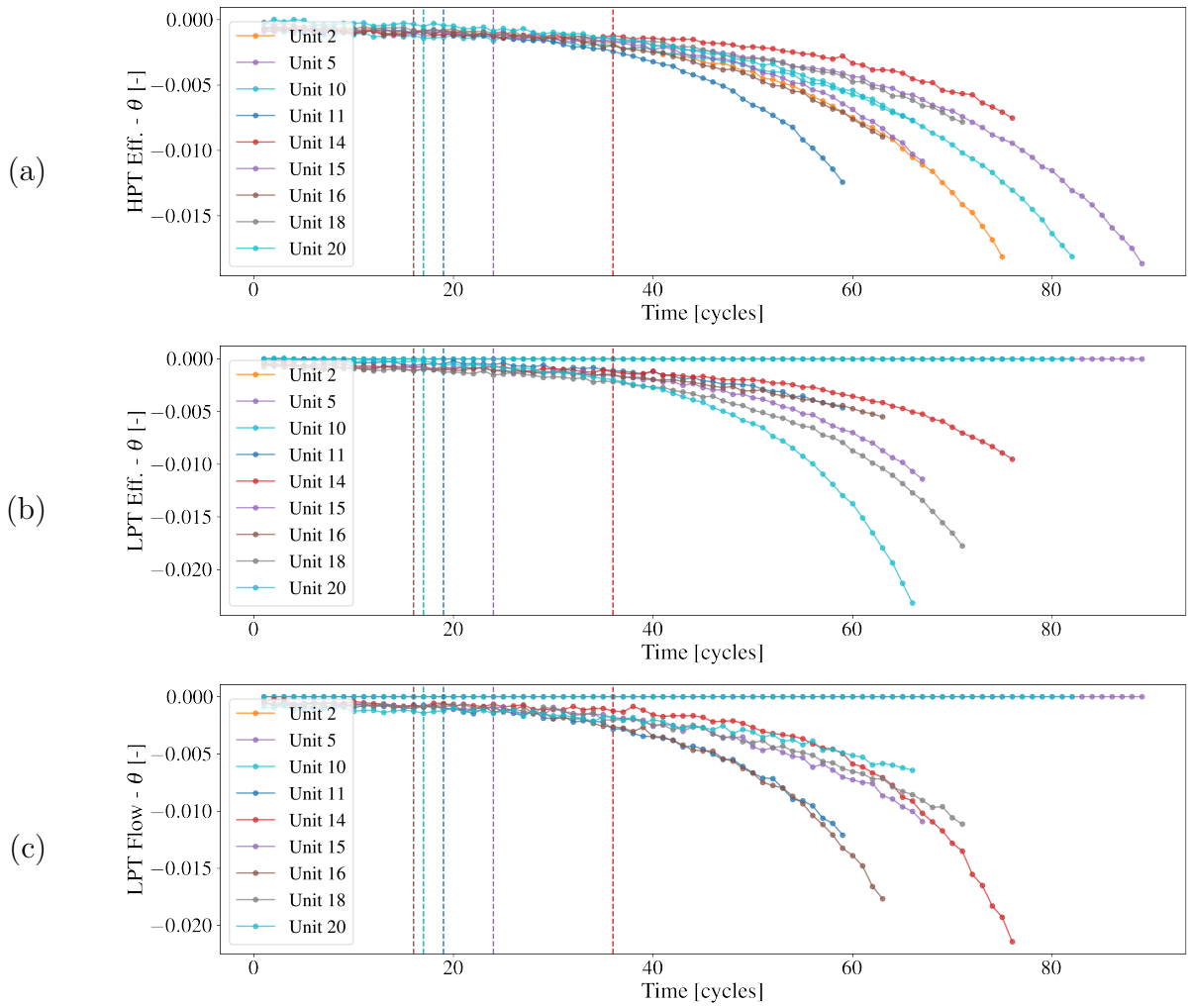
In this application example, the units have been subjected to high and low-pressure turbine failure modes with an initial condition of random deterioration of about 10% of the health index implicit in the simulator. Table 5 details the failure modes for each unit and provides additional information on the number of samples, the transition time to abnormality (t_s), and end of life (t_{eol}) in cycles. Figure 12 details the trajectory imposed on the flow and efficiency modifiers for the tested units and, finally, Table 12 in Appendix A lists the description of the variables that make up the dataset that feeds the procedure.

Table 5: Information about subset samples of each unit. t_s represents the real transition time between normal and degraded conditions.

Dataset Fraction	Unit (u)	Samples	t_s (Cycles)	t_{eol} (Cycles)	Failure mode
Training	2	0.085M	17	75	HPT
	5	0.103M	17	89	HPT
	10	0.095M	17	82	HPT
	16	0.077M	16	63	HPT+LPT
	18	0.089M	17	71	HPT+LPT
	20	0.077M	17	66	HPT+LPT
Test	11	0.066M	19	59	HPT+LPT
	14	0.016M	36	76	HPT+LPT
	15	0.043M	24	67	HPT+LPT

Source: Adapted from Chao et al. (2022)

Figure 12: Evolution of flow and efficiency modifier trajectories over time. Degradation is introduced into the system in the timestep indicated by the vertical dashed lines. Therefore, the modulation of these inputs induces a response in the sensors array.



Source: Chao et al. (2021)

4.1.1 Fixed hyperparameters combination and comparison with baseline model

This application example follows the framework with sets of hyperparameters and fixed neural network architecture. The feature space is composed by 18 variables, which are the same condition monitoring signals used by Chao et al. (2022). Besides that, a detailed description of the CMAPSS simulator variables can be found in (FREDERICK; DECASTRO; LITT, 2007).

The autoencoder models are subject to a validation procedure that consists of two steps: the first one is to evaluate whether its performance, through the analysis of the metrics presented in Section 4.1.2.1, surpasses that of a simplified baseline model, which does not use deep learning, and the second is to compare their results with alternatives presented in the literature that employ similar techniques and databases.

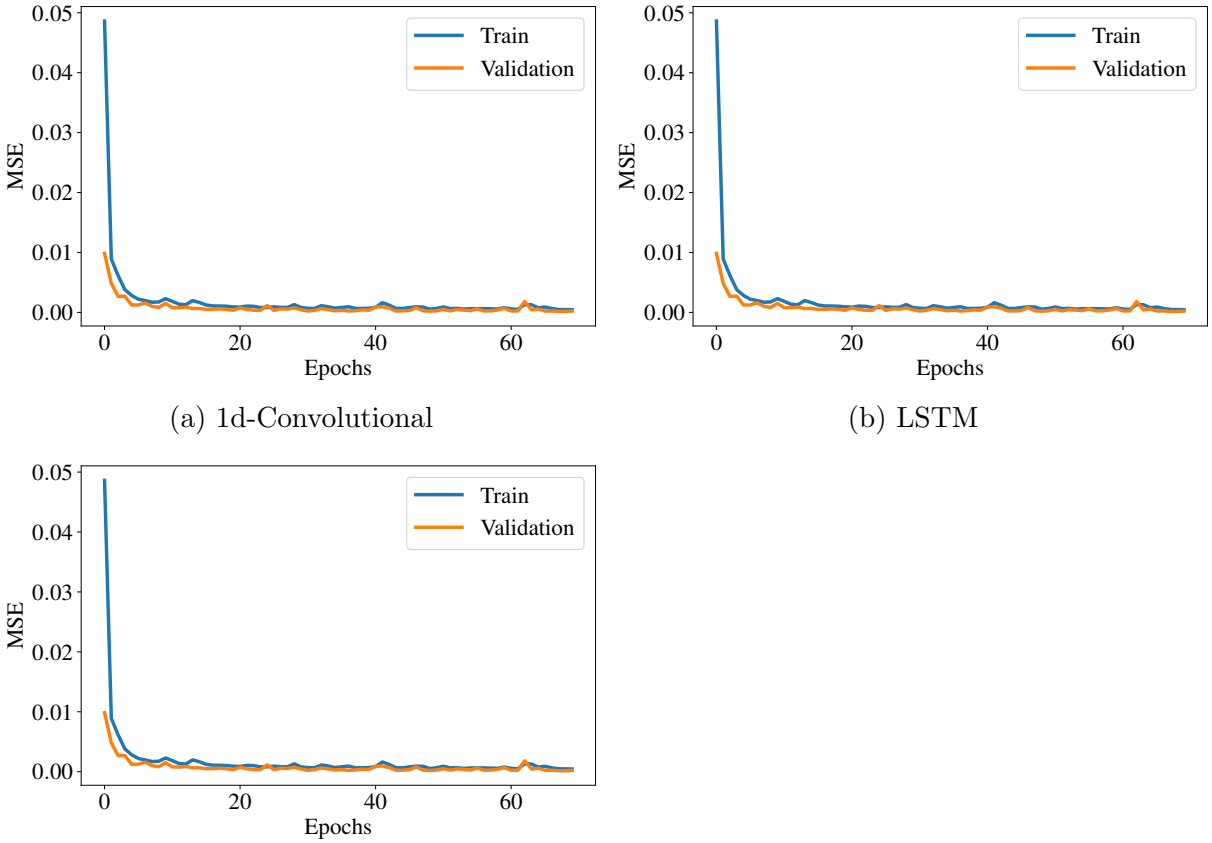
The baseline model is built from a simple regression extrapolation procedure of the pre-processed original inputs – sensors – of the database, following the sequence of steps: downsampling at a rate of 1 sample every 200, without crossing the limits of operational cycles, and later smoothing by simple moving average of size 500, so that the amount of samples of this model and the one submitted for validation is similar. Its output is then applied in Algorithm 1 of the methodology and performance is evaluated with the metrics of Section 3.1.4.

The MSE loss convergence during the networks' training progression is shown in Figure 13 and the progression of RUL estimations over the course of the operation of the units is shown in Figures 14 and 15 The time instant t, x-axis, is normalized in relation to the total life (t_{eol}) of the engines and is interpreted as a percentage (0 - 100%) of t_{eol} or as normalized cycles. The y axis indicates the predicted RUL at instant t, also expressed as a percentage of t_{eol} and the orange dashed line, the real value of the RUL, that is $t - t_{eol}$, at that instant. It is noted that the beginning of the forecast differs from the units since it is directly related to the abnormality detection capacity, which is made by a criterion similar to that used by Rosa et al. (2022b), wherein there is a difference in a consecutive set of points of the maximum reconstruction error between the samples in NOC.

For all the analyzed models, the time of the first prediction (t_{fpt}) occurred after half of the total life of the engines. From 50 to 65% there is a region of instability in the forecasts, in which there are remaining life estimates that exceed the value of t_{eol} near of 100% or underestimate it close to 1 %. This is because the deterioration trends are incipient and have a low rate of change, which causes difficulties for the algorithm to decide which of

the curves is the most appropriate, as some have a very similar fit condition. After 70% of the t_{eol} , a stable convergence zone is formed and the adherence of the projections to the real RUL curve gradually improves up to 100%, which is the desired behavior. Compared to the baseline model, the proposed models advance to a stable condition much earlier (65%) than the baseline (80%).

Figure 13: MSE loss convergence during the networks training progression. An epoch stands for a training cycle when the entire training dataset is used.



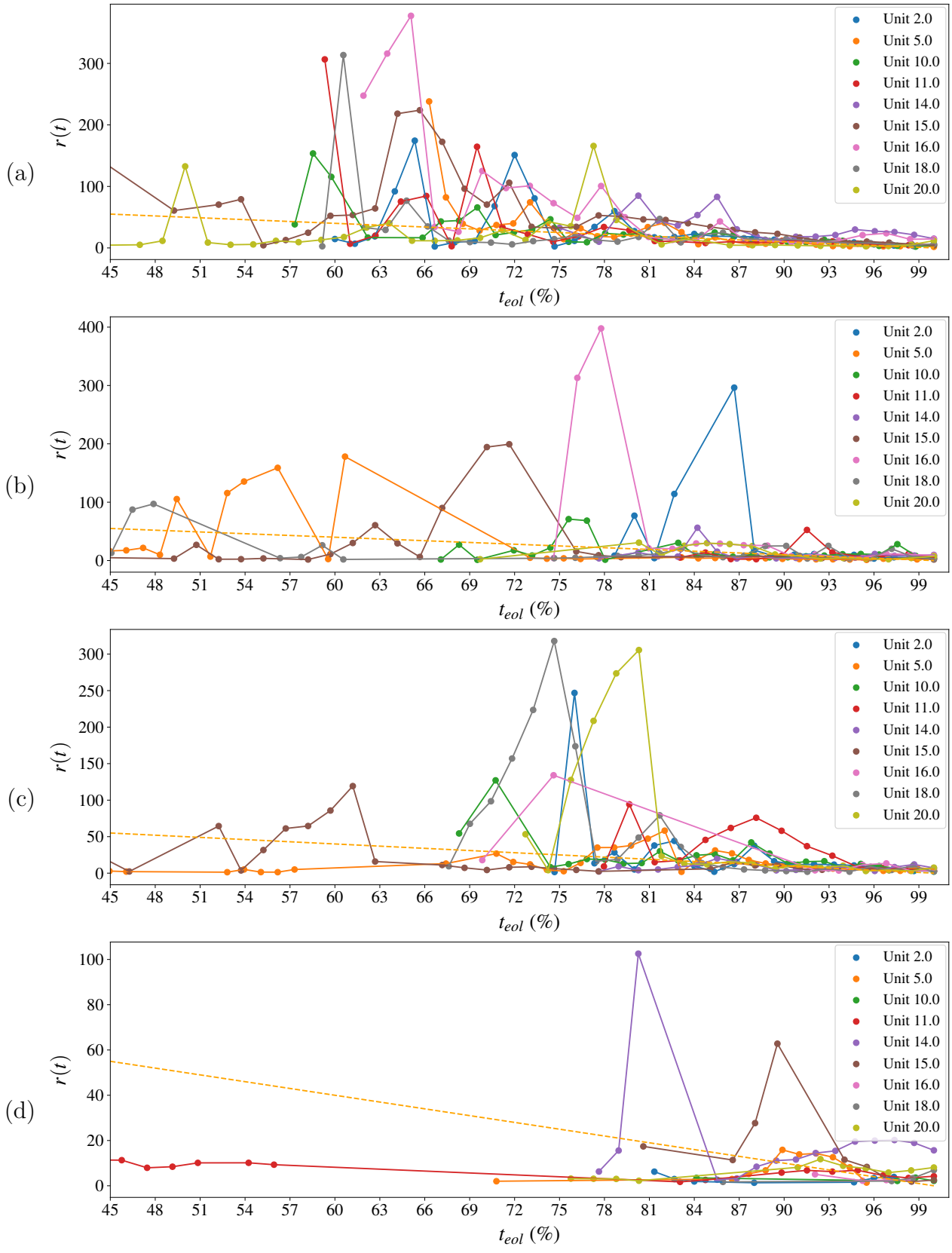
(a) 1d-Convolutional

(b) LSTM

(c) MLP

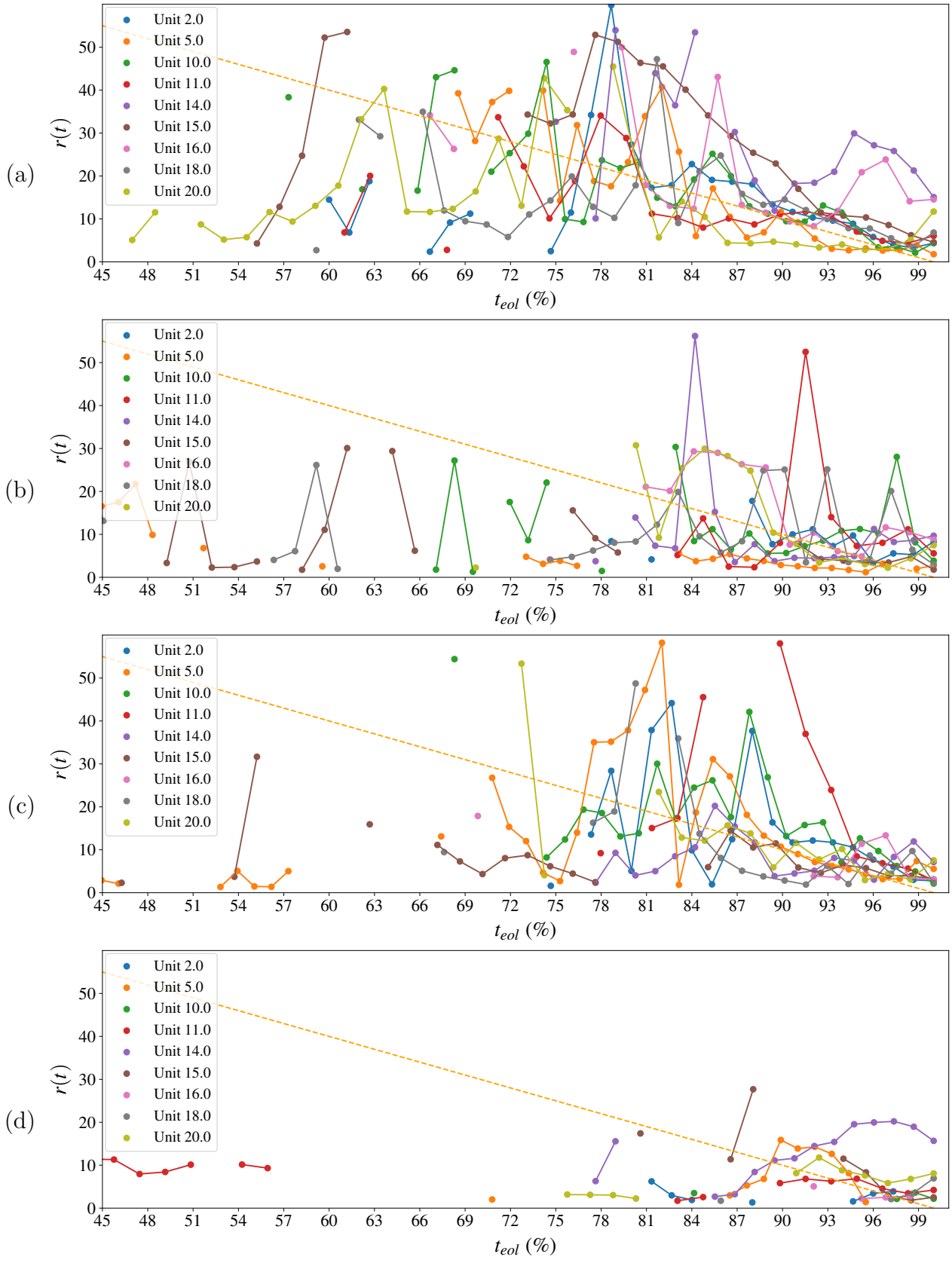
Source: Author

Figure 14: Progression of predictions over units' operational time - $r(t)\,\mathrm{x}\,t_{eol}$ (%) - for the autoencoder reconstruction error extrapolation method and baseline model. The dashed orange line means the ground truth RUL. (a) 1d-convolutional, (b) LSTM, (c) MLP and (d) Baseline.



Source: Author

Figure 15: Progression of predictions over units' operational time - $r(t)$ x t_i (t_{eol}) - for the autoencoder reconstruction error extrapolation method and baseline model. Close-up view with error above being 60% t_{eol} suppressed. The dashed orange line means the ground truth RUL. (a) 1d-convolutional, (b) LSTM, (c) MLP and (d) Baseline.



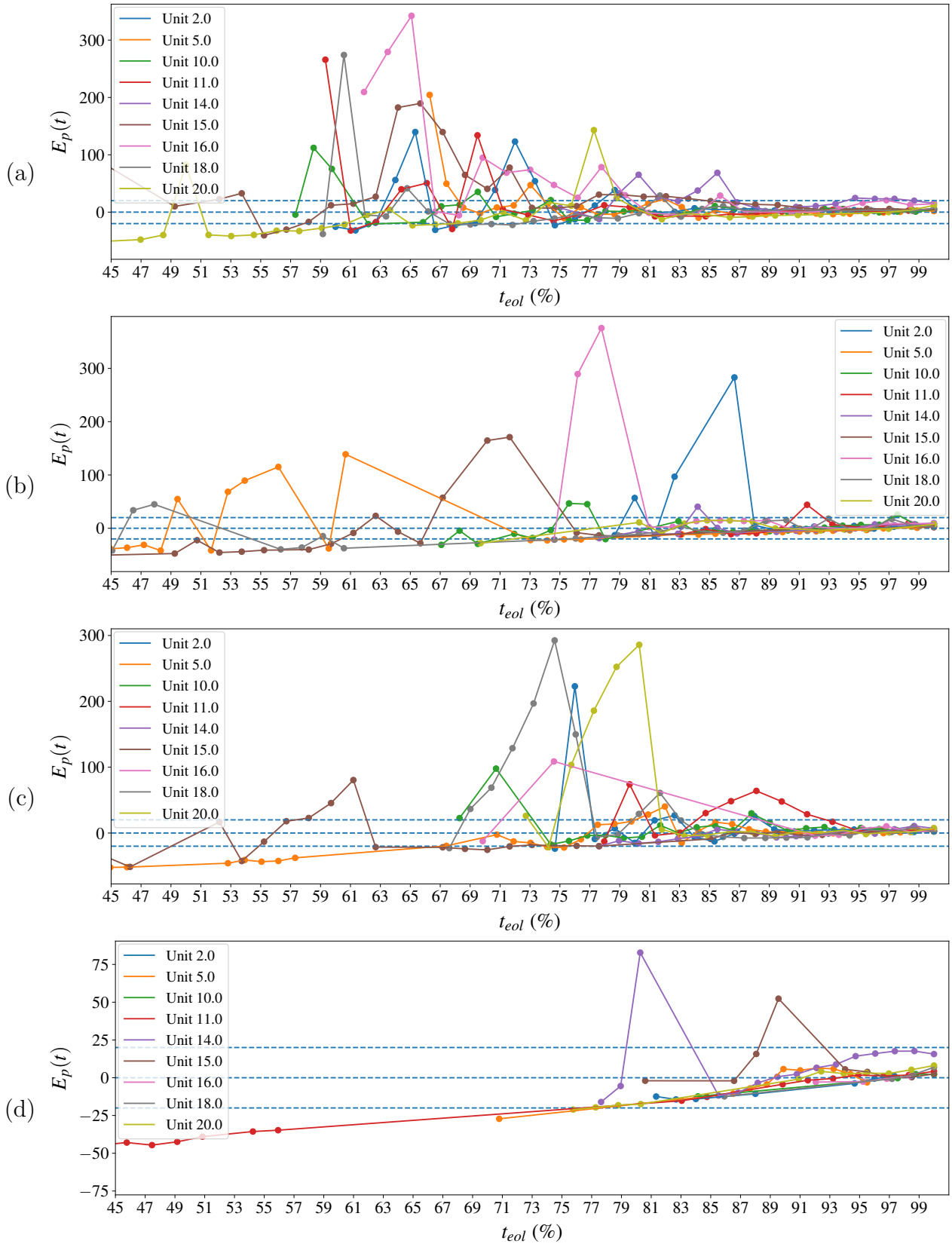
Source: Author

From the three models tested, Conv-1d showed the best result in terms of advancing convergence to the actual prognostic result for all units. It can be seen from Figure 14 that it is the model with the most anticipated first average prediction time of all the units and adheres to the reference line of progression of the RUL in about 75% of the t_{eol}. The MLP model visually manifested a behavior similar to the convolutional one and also presented a zone of forecast instability with high fractional RMSE, but with time stamping metrics (t_{fpt}, $H_{\top(5)}$ and $H_{\top(20)}$) later compared to the second. The LSTM model did not show a concentrated region of large prediction error like the previous two, but it did show sparse peaks of high error for two or three cycles in units 2, 16, 15 and 5. Although it may seem that the LSTM provides more stable predictions, in fact, gaps in the forecasts may be occurring, especially in the region of 60-75 % t_{eol}, in which large magnitude discrepancies are suppressed by the restriction of the algorithm to disregard RUL estimates, if $r^*(t) + t$ exceeds t_{eol} above 300 cycles.

For a moving average subsequence of n = 500, it can be seen that the three autoencoder models outperform the Baseline, which starts to provide consistent forecasts after 80 normalized cycles have elapsed. Perhaps an increase in the time window of the moving average could proportionate a positive impact, especially on the base model, as it benefits the most from signal attenuation in regions of instability. However, increasing n reduces the number of samples of each unit available for curve fitting in the prediction algorithm so that the RUL of some units arranged in Table 5 could not be calculated.

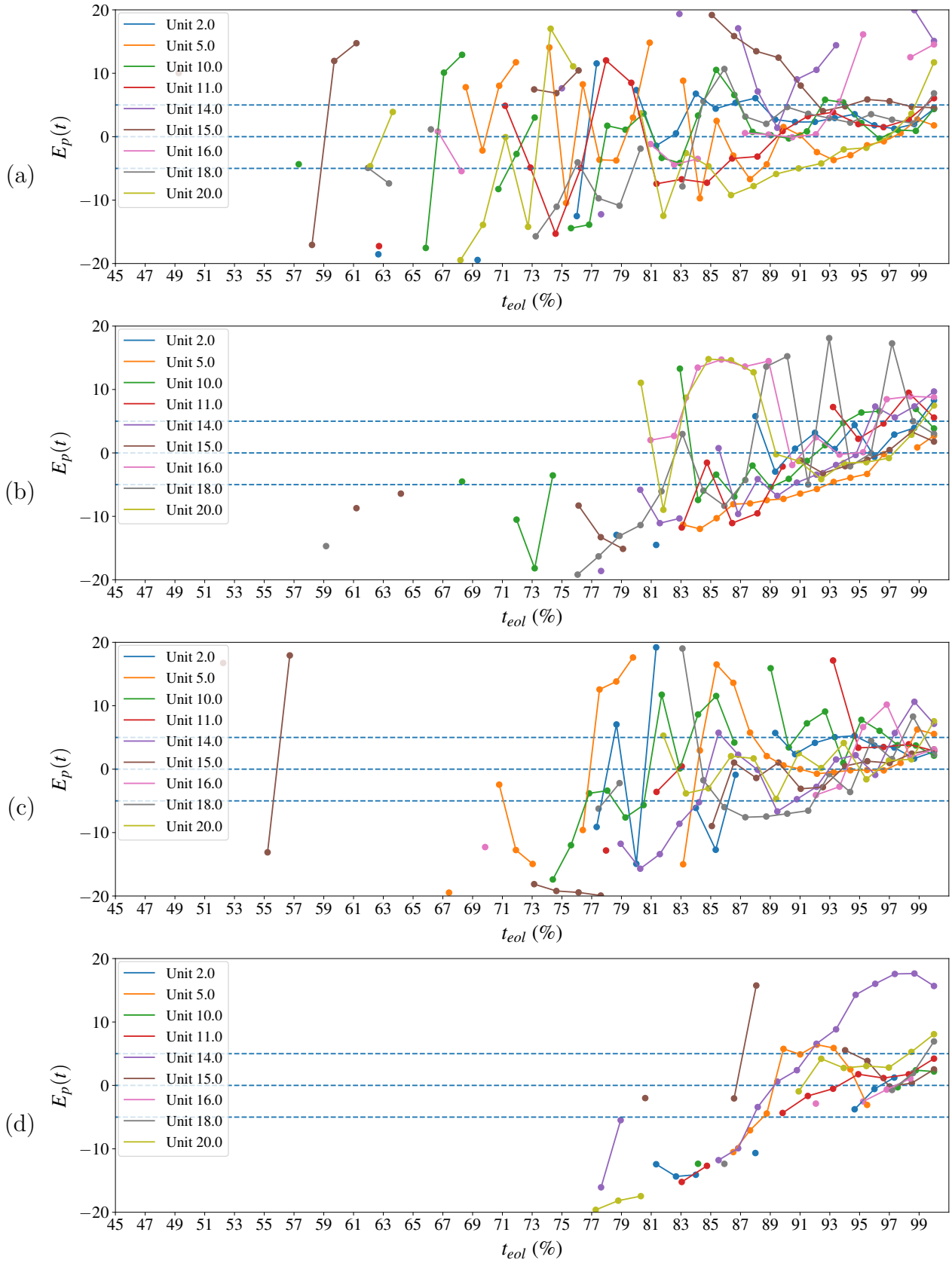
The difference between the forecast and the actual value of the RUL, also expressed as a percentage of the t_{eol} , is shown in Figures 16 and 17. The blue dashed lines indicate 20% error limits, in Figure 16, and 5% error in Figure 17, taken as a reference for calculating the prognostic horizon. The proposed method manages to keep the estimates within the error margin of +/- 20 % t_{eol}, but has complications in meeting the goal of +/-5% t_{eol}, with only a few units achieving this result even after 80% of the machine's life. There are two possible reasons for this answer: the first one, mentioned above, is the absence of global tuning of the model, including the neural architecture, which is not at its optimal performance in terms of training with NOC samples; the second one is the uncertainty regarding the choice of the error threshold for the prognosis, which can affect on the estimates above what was expected.

Figure 16: Progression of the prediction error relative to the total life of the asset, $E_p = 100 * (r^*(t) – r(t))/t_{eol}$ over the time of operation of the units for the reconstruction error extrapolation and baseline methods. The prognostic horizon of 20 % is represented by the blue dashed lines. (a) 1d-convolutional, (b) LSTM, (c) MLP and (d) Baseline.



Source: Author

Figure 17: Progression of the prediction error relative to the total life of the asset, $E_p = 100 * (r^*(t) – r(t))/t_{eol}$, over the time of operation of the units for the reconstruction error extrapolation and baseline methods. The prognostic horizon of 5 % is represented by the blue dashed lines. (a) 1d-convolutional, (b) LSTM, (c) MLP and (d) Baseline.



Source: Author

The summary of the results obtained for the values of the performance metrics is presented in Table 6, while Table 7, presents all the results organized by unit. Both tables show the $RMSE$, fractional $RMSE$'s $L1$, $L2$ and $L3$, time of the first prediction (t_{fpt}), the ns (Equation 3.6) divided by the total number of estimates, cumulative relative accuracy and prognostic horizons for 5 and 20 % errors. It should be noted that $L1$, $L2$ and $L3$ stand for the RMSE fraction only for samples inside the first, second, and third thirds, respectively, of the normalized t_{eol} second half.

Table 6: Mean and deviation of the performance metrics computed for each model and autoencoders overall. Population size is equal to the total amount of Units in Table 5.

Performance Metrics	Conv-1d		MLP		LSTM		Conv-1d, MLP, and LSTM Overall		Baseline	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
$RMSE$	49.702	23.942	44.116	30.256	42.505	36.587	45.441	6.322	**14.702**	12.102
$RMSE_{L1}$	122.926	81.626	**44.815**	5.514	49.569	17.053	72.437	41.02	52.499	-
$RMSE_{L2}$	35.958	15.82	63.927	52.562	52.505	66.543	50.797	26.198	**20.968**	15.921
$RMSE_{L3}$	8.668	7.929	9.312	9.856	18.267	27.008	12.082	10.503	**7.964**	5.549
t_{fpt}	**57.627**	11.532	65.039	15.695	62.454	19.762	61.707	4.115	75.633	19.729
ns	**0.403**	0.002	0.402	0.002	0.4	0.002	0.402	0	0.4	0.002
CRA	-0.311	0.912	-0.488	0.708	-0.482	0.785	-0.427	0.103	**0.006**	0.633
$H_{\top(5)}$	**11.767**	6.123	8.205	7.847	4.169	4.69	8.047	1.581	4.724	2.984
$H_{\top(20)}$	**20.694**	5.336	14.735	4.75	12.072	8.399	15.834	1.959	6.643	4.27

*Highlighted in bold, the best result in each row.

There is no expressive gain in $RMSE$ of the proposed model (45,441) when compared to the Baseline (14,702, Table 6) due to the rough projections made at the beginning of the degradation process. When these prognostic samples are disregarded, it is possible to notice a performance gain for this metric, which is expressive from the third third ($L3$) and improves in the proximity of t_{eol}, therefore quantitatively corroborating that the proposed model advances to a state of convergence zone before the baseline.

A inspection in Table 6 reveals that the overall $RMSE$ is lower than the Baseline model. The reasons are that the Baseline model produced less and late estimates in comparison to the autoencoders, as can be viewed in Figure 14 and Figure 15, and when closer to the end of life, the predictions errors tend to be smaller due to the presence of more information about the pronounced degradation. The models start to equate in performance as there is an approximation to the stable convergence zone, and there is a slight divergence between the $RMSE_{L3}$ values. Although Baseline also has lower $RMSE_{L3}$, it should be noticed that it has performed fewer predictions even in that region – see Figure 14 and Figure 17. The prognostic horizon is certainly greater for the autoencoders-based solutions, highlighting the Conv-1d, which has the earlier t_{ftp}, so

a correlation with the $H_{\top(C)}$ has already been expected. The difference t_{ftp}–$H_{\top(C)}$ could be interpreted as a latency of the model in archive an acceptable error margin.

Moreover, ns, another error evaluation metric, differed from the $RMSE$'s outcomes by showing a similar quantity overall. This fact is justifiable because, even though the models have differed significantly in global accuracy, all of them displayed a greater tendency to overestimate predictions, which is penalized by this metric. CRA, in turn, follows the $RMSE$ behavior, as they are almost analogous measurements when a linear weighting $w(x) = x$ (Equation 3.12) is taken.

Generally, the proposed autoencoder models are more stable than the Baseline model, detect abnormalities earlier, and enter a region of stable convergence earlier. They manage to meet the margin of error requirement below 20% of t_{eol} for at least a fourth of the unit's life but struggle to meet the requirement of a 5% forecast horizon.

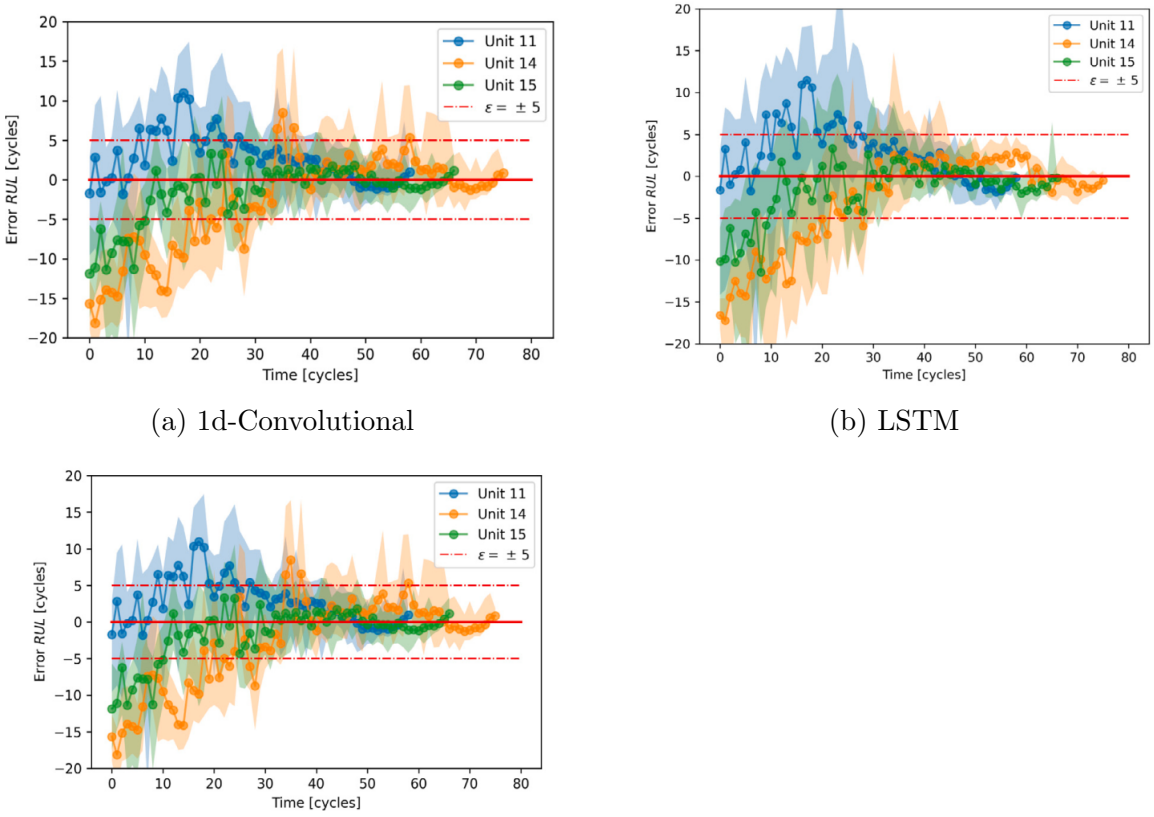
Finally, the comparison with the literature is based on the publication by Chao et al. (2022), who built deep learning models to estimate the RUL also on the CMAPSS-2 basis. This comparison aims to verify if the framework exhibits coherent behavior for the predictions over time. It is made by the qualitative inspection of the prediction errors progression, see Figure 16 and Figure 17, which is also plotted by these authors for the same three kinds of layers used in this study. Moreover, some performance measurements taken in this work are compared with the results obtained by the referred author. They are RMSE and prognostic horizon.

The presented models could not overcome the data-oriented arrangements programmed by Chao et al. (2022), nor is this the intention, as they use supervised learning, thus mapping the channel signature throughout the degradation evolution and not only in the NOC. Even if it is not possible to exceed this author in performance, it is important to note that there is a great proximity between the mean squared error values for the stable convergence zone ($RMSE_{L3}$).

Figure 18 similarly illustrates the progression of estimates from RUL to t_{eol} for a one-dimensional convolutional, LSTM and MLP models tested by Chao et al. (2022). There is a great similarity between the behavior of the operating time forecasts plotted in Figure 18 with the one shown in this work. Greater uncertainty is also demonstrated at the beginning of the forecasting process and it is gradually reduced until t_{eol}. It is observed that the use of a supervised technique allows a t_{fpt} very close to the beginning of the unit's life and that the supervised method purely derived from ANN can make inferences almost in real time after being trained and by a new sample generated, without

the computational cost imposed by curve fitting. On the other hand, supervised learning techniques tend to be more specific to the application – failure mode – and have a shorter lifespan, requiring retraining to adapt to changes in the operating equipment.

Figure 18: Prediction errors for RUL of the supervised models from Chao et al. (2022).



(a) 1d-Convolutional

(b) LSTM

(c) MLP

Source: Chao et al. (2022)

Table 7: Performance metrics for each unit remaining life evaluation of the tested models. L_1, L_2,L_3 stands for the $RMSE$ fraction only for samples inside the first, secord and third thirds, respectively, of the normalized t_{eol} second half. ns is an adaptation of the s-score - Equation 3.6 - that considers normalized RUL.

Model	Unit	$RMSE$	$RMSE_{L1}$	$RMSE_{L2}$	$RMSE_{L3}$	t_{fpt}	ns	CRA	$H_{\top(5)}$	$H_{\top(20)}$
	2.0	28.174	12.882	14.076	8.793	61.333	0.402	-0.052	4.000	9.333
	5.0	39.661	9.798	7.560	1.299	64.045	0.403	0.408	12.360	33.708
	10.0	11.883	8.405	8.760	2.491	57.317	0.405	0.500	9.756	42.683
	11.0	42.548	6.922	5.427	6.443	59.322	0.402	0.095	1.695	30.508
Conv-1d	14.0	91.845	91.845	35.161	48.548	97.368	0.407	-31.796	0.000	0.000
	15.0	70.579	37.695	21.768	14.243	52.239	0.404	-0.791	0.000	19.403
	16.0	82.639	83.927	6.662	3.703	47.619	0.404	-0.417	11.111	23.810
	18.0	58.992	10.021	5.521	2.912	30.986	0.400	0.260	32.394	32.394
	20.0	24.333	16.163	8.475	3.301	31.818	0.398	0.430	15.152	51.515
	2.0	28.174	12.882	14.076	8.793	61.333	0.402	-0.052	4.000	9.333
	5.0	39.661	9.798	7.560	1.299	64.045	0.403	0.408	12.360	33.708
	10.0	11.883	8.405	8.760	2.491	57.317	0.405	0.500	9.756	42.683
	11.0	42.548	6.922	5.427	6.443	59.322	0.402	0.095	1.695	30.508
MLP	14.0	91.845	91.845	35.161	48.548	97.368	0.407	-31.796	0.000	0.000
	15.0	70.579	37.695	21.768	14.243	52.239	0.404	-0.791	0.000	19.403
	16.0	82.639	83.927	6.662	3.703	47.619	0.404	-0.417	11.111	23.810
	18.0	58.992	10.021	5.521	2.912	30.986	0.400	0.260	32.394	32.394
	20.0	24.333	16.163	8.475	3.301	31.818	0.398	0.430	15.152	51.515
	2.0	28.174	12.882	14.076	8.793	61.333	0.402	-0.052	4.000	9.333
	5.0	39.661	9.798	7.560	1.299	64.045	0.403	0.408	12.360	33.708
	10.0	11.883	8.405	8.760	2.491	57.317	0.405	0.500	9.756	42.683
	11.0	42.548	6.922	5.427	6.443	59.322	0.402	0.095	1.695	30.508
LSTM	14.0	91.845	91.845	35.161	48.548	97.368	0.407	-31.796	0.000	0.000
	15.0	70.579	37.695	21.768	14.243	52.239	0.404	-0.791	0.000	19.403
	16.0	82.639	83.927	6.662	3.703	47.619	0.404	-0.417	11.111	23.810
	18.0	58.992	10.021	5.521	2.912	30.986	0.400	0.260	32.394	32.394
	20.0	24.333	16.163	8.475	3.301	31.818	0.398	0.430	15.152	51.515
	2.0	28.174	12.882	14.076	8.793	61.333	0.402	-0.052	4.000	9.333
	5.0	39.661	9.798	7.560	1.299	64.045	0.403	0.408	12.360	33.708
	10.0	11.883	8.405	8.760	2.491	57.317	0.405	0.500	9.756	42.683
	11.0	42.548	6.922	5.427	6.443	59.322	0.402	0.095	1.695	30.508
Baseline	14.0	91.845	91.845	35.161	48.548	97.368	0.407	-31.796	0.000	0.000
	15.0	70.579	37.695	21.768	14.243	52.239	0.404	-0.791	0.000	19.403
	16.0	82.639	83.927	6.662	3.703	47.619	0.404	-0.417	11.111	23.810
	18.0	58.992	10.021	5.521	2.912	30.986	0.400	0.260	32.394	32.394
	20.0	24.333	16.163	8.475	3.301	31.818	0.398	0.430	15.152	51.515

4.1.2 Grid-search experiment

The grid-search experiment was carried out to promote a more in-depth sensitivity study on the hyperparametric space of the created framework. In total, 931 neural networks were trained, culminating in 2870 models of detection and prognosis. Each one was simulated on the RTF's trajectories of the CMAPSS base as in the previous application example. Thus, there was a 89.69 % utilization from the planned grid with 3200 combinations. Table 8 illustrates the proportion of tested models for each of the hyperparameters categories, with each line representing the total number of configuration samples that contain the indicated attribute. It also provides information about the elementary constitution of the grid.

It is emphasized that the investigations are conducted in the neighborhood of a reference point relative to the space of combinations of hyperparameters defined based on setups from previous studies conducted by Chao et al. (2022) and Rosa et al. (2022b). It is analogous to the fixed combination used in the first part of the application example. Hence, the other grid values derive from this point at an appropriate discretization step so that it is feasible to train all grid components in a timely manner and there are no interposed parameters that invalidate the models.

The code was executed on a machine equipped with an RTX 3060 video card with 12 GB VRAM memory, Intel(R) Core(TM) i7-10700 processor, and 16 RAM memory in an approximate computational effort of 15 days of uninterrupted operation.

The analysis was divided into three main parts according to the emphasis and graphic resources used: a global analysis of the metric space response by hyperparameter class, temporal profiles in normalized cycles for detection and prognostic activities, and finally correlation and clustering of metrics for composition of objective functions and ranking of samples. In this way, it is possible to have an overview of the performance and the temporal progression of the tasks, which makes it possible to identify locally the differences in performance and spurizations. In the end, attest to the consistency of the chosen metric space and its importance in the selection of optimal examples, thus fulfilling the purpose of the tuning.

The metric set is chosen by the considerations about the individual quality of the metrics made in Table 4, thus the group formed can properly attest to the functionality of the models produced. Therefore, a consistent metric space is one that is able to measure the detectability, extensibility and predictability requirements in a lean way, that is, with the smallest possible number of elements. More detailed definitions for these concepts can

be found in Table 9 in Section 4.1.2.3. In addition, desirable but non-eliminating characteristics include linear independence between metrics, dimensionless, bounded domain, a minimum amount of auxiliary parameters and easy graphical representation.

Table 8: Proportions of the tested models for each kind of hyperparameter entry.

Hyperparameter	Entry	Quantity	%
Type	Conv1d	1417	49.37
	LSTM	798	27.8
	MLP	655	22.82
Subsequence Size	60.0	468.0	16.31
	200.0	529.0	18.43
	300.0	515.0	17.94
	400.0	511.0	17.8
	500.0	461.0	16.06
	700.0	386.0	13.45
Neurons	(16, 16)	163	5.68
	(16, 4)	510	17.77
	(16, 8)	517	18.01
	(32, 16)	358	12.47
	(32, 32)	165	5.75
	(32, 8)	162	5.64
	(64, 16)	161	5.61
	(64, 32)	153	5.33
	(64, 64)	164	5.71
	(8, 4)	517	18.01
Kernels	(10, 10)	355	25.05
	(10, 5)	357	25.19
	(15, 10)	357	25.19
	(25, 10)	348	24.56
Overlapping	0.0	718.0	25.02
	20.0	735.0	25.61
	30.0	742.0	25.85
	50.0	675.0	23.52
Moving Average Window	200.0	717.0	24.98
	400.0	756.0	26.34
	1000.0	760.0	26.48
	2000.0	637.0	22.2

4.1.2.1 Analysis of the metric space global response

The influence of the type of layer used, highlighted in Figure 19, manifests itself on the mean squared error values as follows: convolutional layers exhibit a higher $RMSE$,

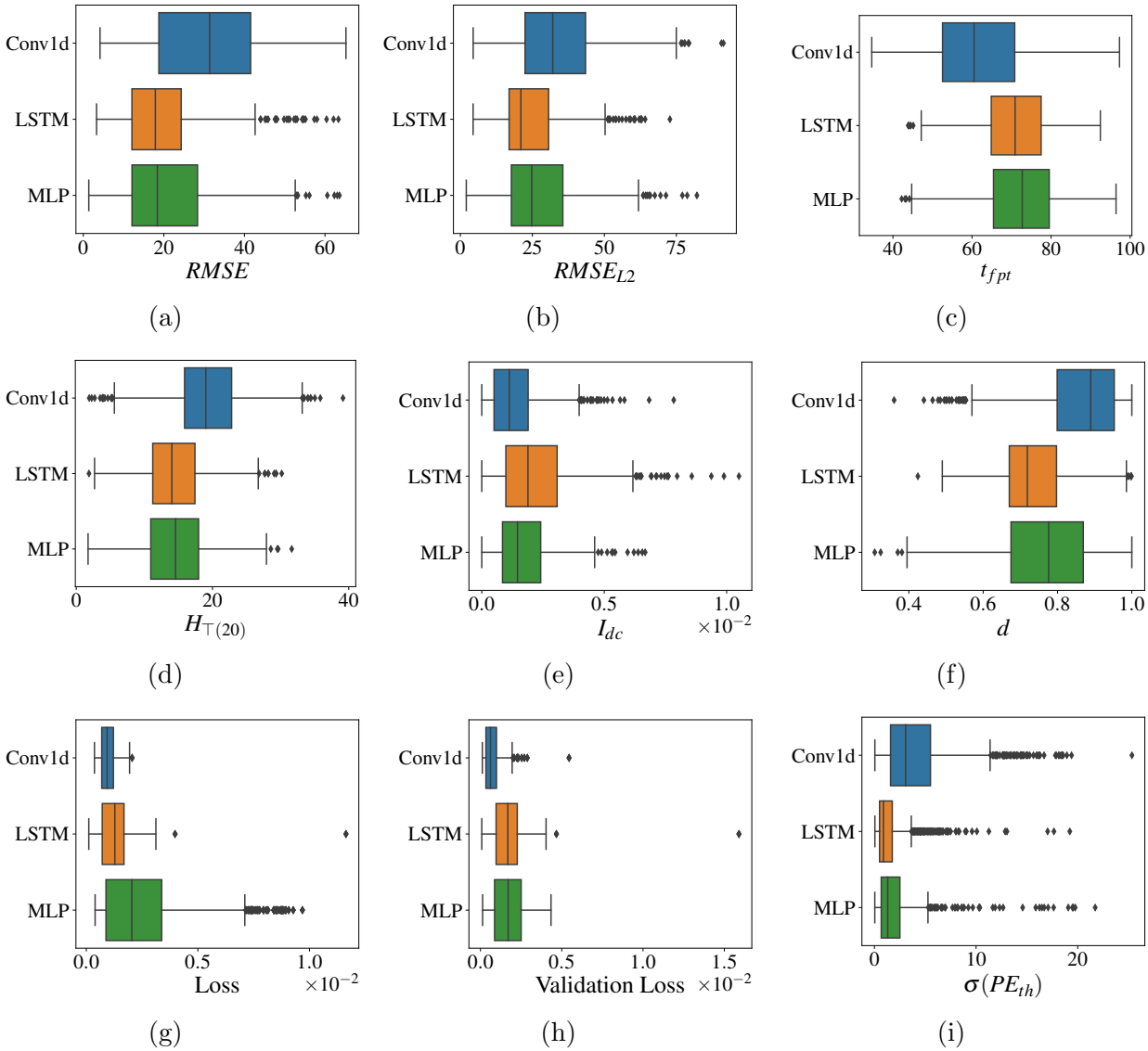
a consequence of a higher $RMSE_{L2}$ at the beginning of the forecast and a lower first prediction time (t_{fpt}), which implies an earlier start of prognosis than the other layers and therefore has a greater amount of embedded uncertainty. However, for estimates made near the end of the asset's life, those with the lowest error ($RMSE_{L3}$) are made by models with convolutional layers. Prognostic horizons track the trend of t_{fpt}, and reveal that Conv1d models manage to reach error control goals faster than the other two. Regarding the discontinuity index, Conv1d and MLP performed better than LSTM, and these two layers are also the ones with the highest detection coverage. Regarding curve fitting, models with Conv1d stand out, both for training and validation samples. On the opposite side are the MLP layers, which have a lower fit for the cost function in the training samples, which was already expected because this is a layer with a simpler operating mechanism. Finally, the variability in the prognostic error threshold is smaller for MLP, LSTM than Conv1d layers. The latter may be more sensitive to local input discrepancies.

The temporal evolution steps - Figure 20 - constituting the searched grid did not influence the precision of the predictions, but significantly affected the detection metrics. Thus, it is possible to see a strong correlation between the continuity index and subsequence size. This is due to the smaller dispersion of punctual errors within the subsequence and the greater proximity in time scale between two consecutive points of reconstruction errors. The detection coverage tends to increase slightly with the increase in the subsequence size, while the false positive decreases.

The number of neurons is a hyperparameter that needs to be analyzed separately for each type of layer since its sensitivity varies according to this detail in the architecture. Its working principle differs in the case of Conv1d and MLP networks, due to the type of connectivity shared with the previous layer and, for LSTM, the term is being used to refer to the number of units - cells - LSTM distributed in parallel. As it is a hyperparameter that integrates the black box of network operation, explanations about existing correlations with other model variables are difficult to provide. It is understood that a greater number of neurons also implies a greater number of trainable parameters and, therefore, demands a greater volume of data to be satisfied.

In the case of Conv1d - Figure 32 in Appendix B -, the number of neurons and the cost for the training set are directly correlated, although the effect for validation is not pronounced in the same way, demonstrating that any increment above the actual one can lead to overfitting. Threshold variability also appears to be a bit sensitive to this element.

Figure 19: Metrical response for types of layers.

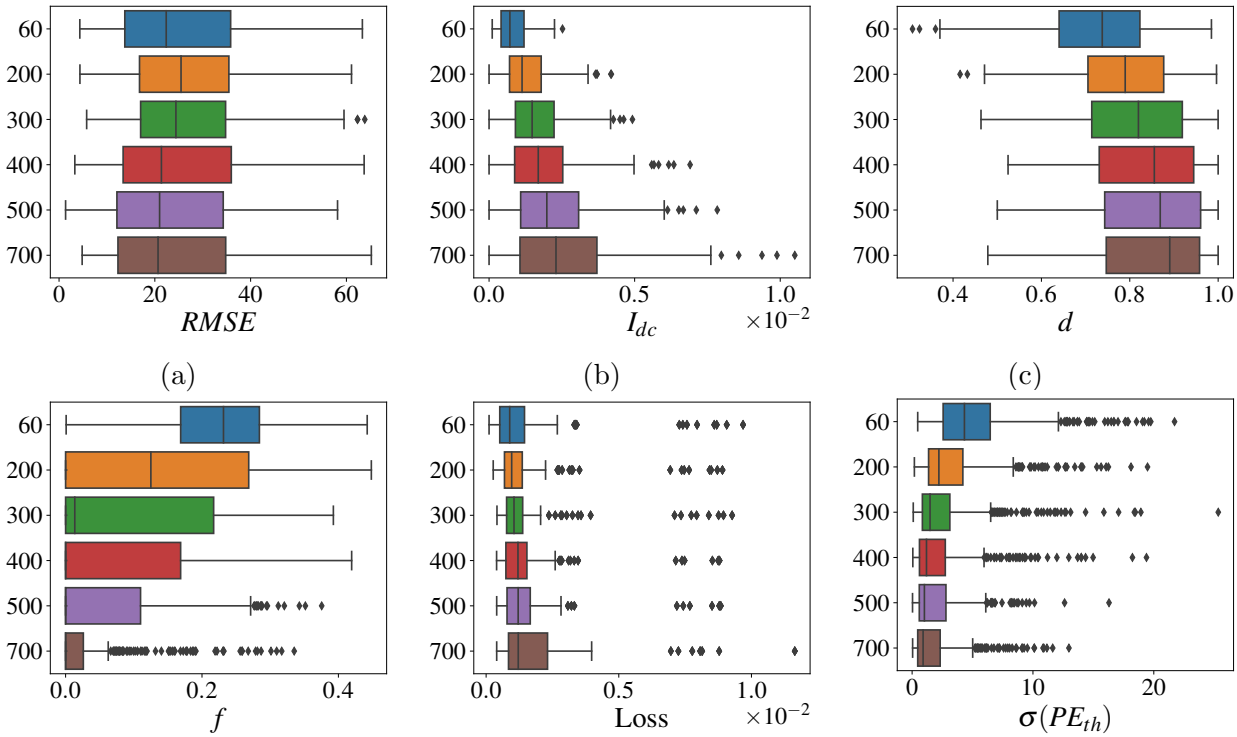


Source: Author

The response of the metric space to the number of neurons in the LSTM-like layers - Figure 33 in Appendix B - was not accentuated to the point that any kind of pattern was noticeable. In general terms, for all possible combinations, the final value of the cost function of the neural networks for the validation and training sets remained invariable, so there will be no propagated instabilities for the reconstruction errors due to the adjustments of the networks to the data in NOC. And thereafter, the following metrics remain unchanged.

Of all networks, MLP was the one that presented the greatest gain in terms of network fit in NOC data ranging from about 0.85e-2 to 0.2e-2 for values of the cost function and 3e- 3 to 0.5e-3 for training data. As shown in Figure 34 in Appendix B, the best fit

Figure 20: Metrical response for subsequence size.



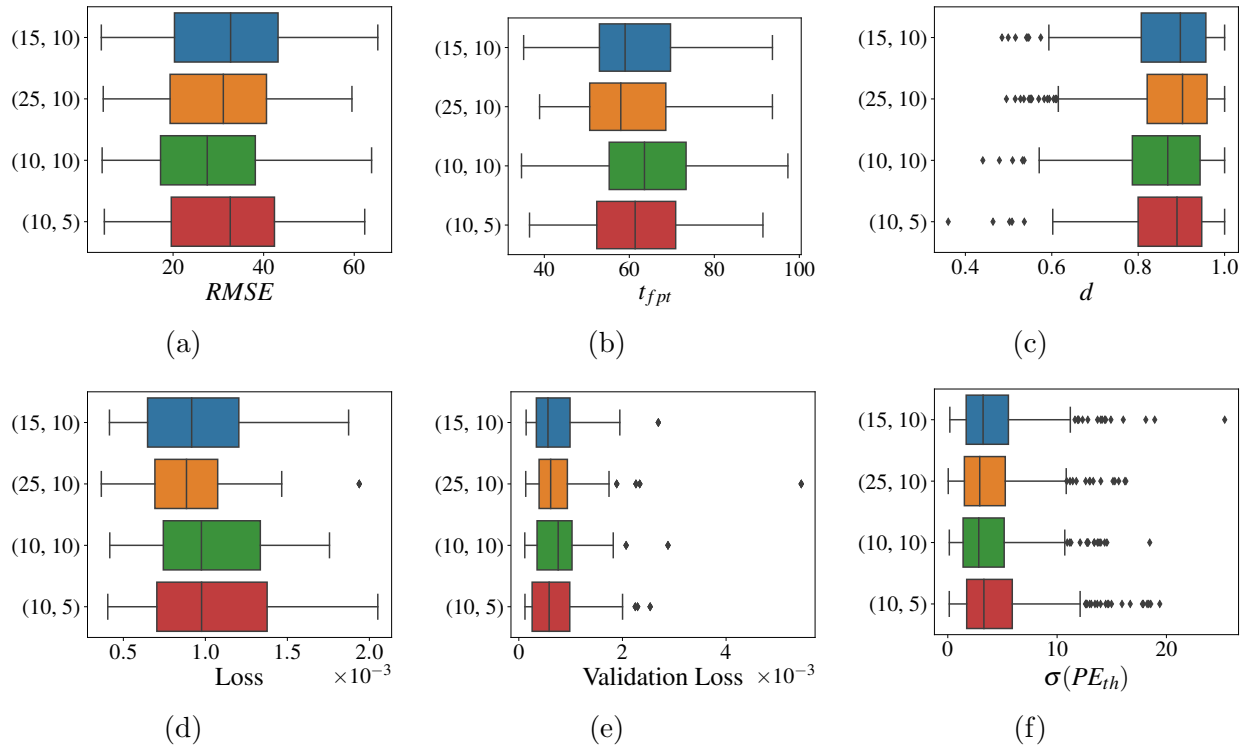
Source: Author

occurred for combinations that resulted in a zero compressibility factor, that is, without dimensionality compression in the autoencoder bottleneck. Although there is a better convergence in the training of the network, this option directly affects the capacity of the network to generalize and infer patterns in inputs that are different from those used in the training. As a result, a deterioration in performance is noted for the MLP models without compressibility factor in other metrics, especially those associated with prediction errors. This attests to the importance of this factor in sizing the autoencoder and, therefore, the autoencoder itself in the development of detection and prognostic methods based on residuals.

The kernel hyperparameter - Figure 21 - concerns the vector of weights that is convoluted to extract the output of features directed to the other layer. It is specific to Conv1d templates. Augmenting the kernel effectively means increasing the number of parameters, making the network more complex. In general, kernel size responds to the scale of the feature being learned, so smaller kernels tend to perceive details, while larger kernels generalize patterns arising from combinations of smaller features. It can be seen from Figure 21 that this variable did not exert a very significant influence on the metric space, so an ablation over a wider range of options would be necessary to understand the

nature of its impact.

Figure 21: Metrical response for different kernels size combinations.

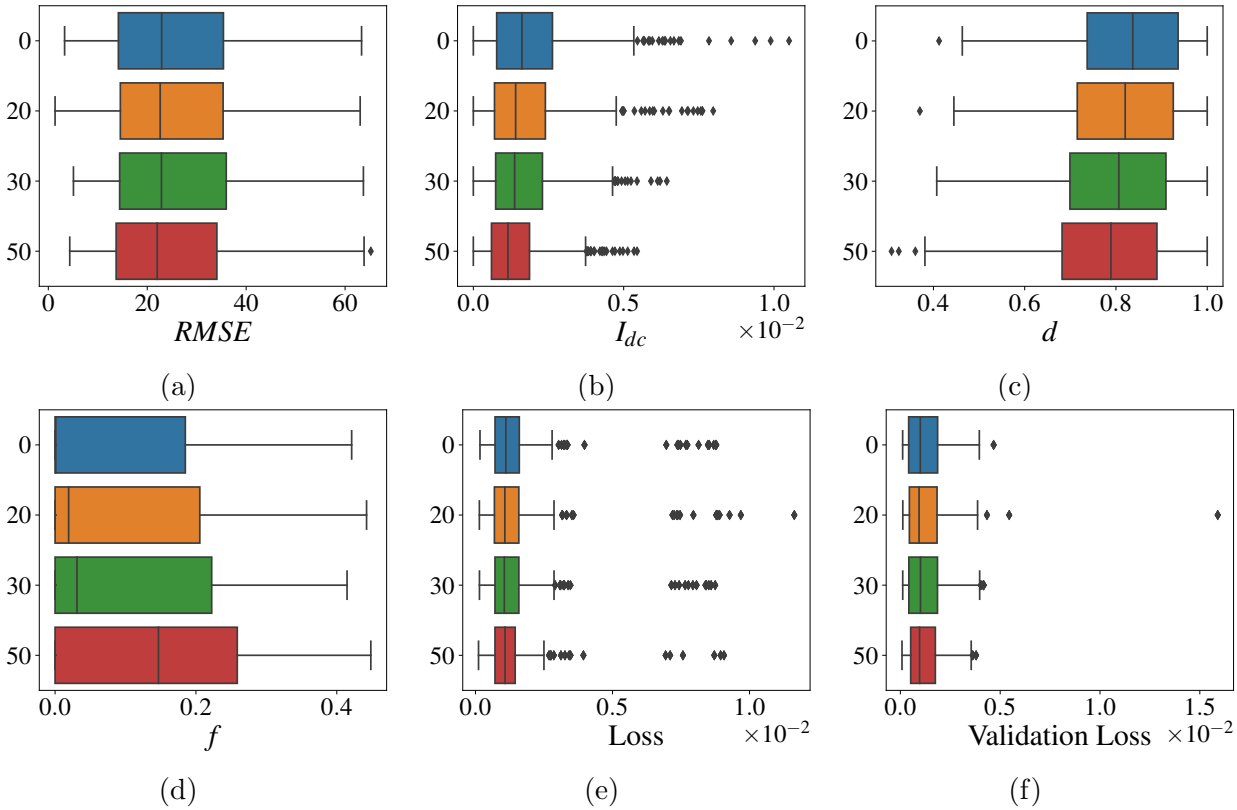


Source: Author

Overlapping subsequences - Figure 22 - acts directly as an outlier attenuator by ensuring a smoother transition between consecutive reconstruction errors. Only overlaps of up to 50% between subsequences were tested, because increments greater than this tended to add too much computational cost and also storage memory cost since the logging produced by the grid-search algorithm predicted the saving of all reconstruction error profiles for each model. For 98% overlapping, for example, a single model would consume about 1.5 GB of disk space. The explosion in size is caused by the growth in the number of samples that approaches in quantity the number of entries in the database of sensor readings. Due to the particularity mentioned above, it was expected that the overlapping would contribute to a smoother transition in the detection of the tipping point of the degraded state. If the discontinuity index is observed, it is verified that this actually occurred. However, greater overlapping penalized detection and false-positive coverage. This hyperparameter proved to be irrelevant for performance in prognosis, as the metrics associated with prediction error remained invariable. A potential justification is that overlapping subsequences does not ultimately affect the rate of change of reconstruction errors, it just inserts more intermediate samples that interpose them and whose values are

diluted by moving averages with window sizes well above the fractions of subsequences.

Figure 22: Metrical response for subsequence overlapping.

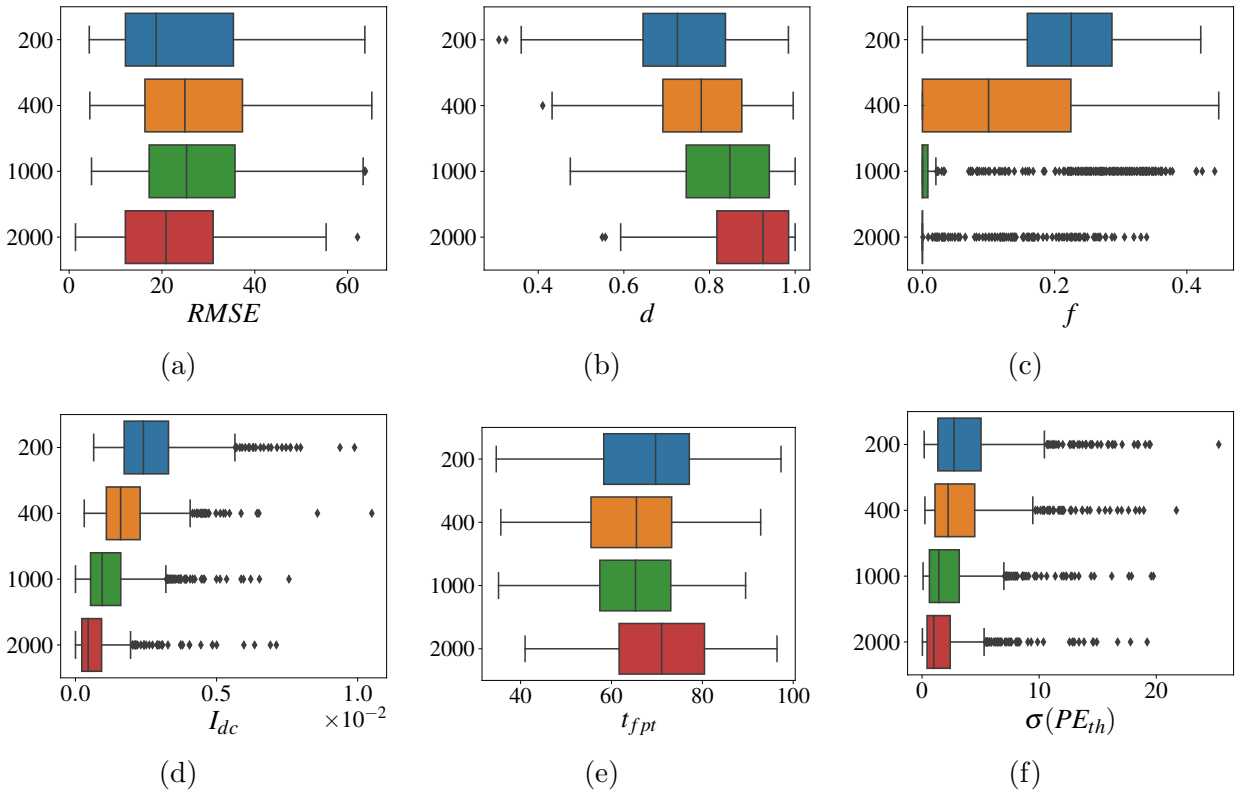


Source: Author

The moving average window – Figure 23 – was the variable that most influenced the others metrics and from which more correlations were perceived for all the models types. Because it is simple, explicit and influences the post-processing of the reconstruction error, its operation is very intuitive. Detection metrics benefit from an increase in the moving average window size. There is a reduction in the number of discontinuities, an increase in detection coverage, and a reduction in false positives. Certainly, the smoothing of reconstruction errors leads to the suppression of outliers and local instabilities and decreases the frequency with which the detection error limit is reached. The existence of a clearer and more coherent transition to the degraded state favors an approximation of the marking of the real tipping point to it, maximizing detection coverage. However, raising the window size too much can lead to a delay in the start of predictions (t_{fpt}), which in turn extends to the prediction horizons. This happens due to the consumption of samples at the beginning of the NOC interval required to compute the first term of the moving average. There are n samples that are no longer visible for detection or curve-fitting, therefore influencing the probability of false-positive occurrence and prognosis. Except for the previously commented behavior, the moving average window size does not

seem to influence the prediction errors, although it also accumulates a positive effect on the variability of the prognosis threshold.

Figure 23: Metrical response for moving average window.



Source: Author

4.1.2.2 Analysis of the time profiles

Time profiles follow the evolution of an indicator from the beginning, or instant of interest, in the life of the asset until its end. They make it possible to highlight nuances that are imperceptible in more general statistical compilations, such as in Section 4.1.2.1, and to verify the proportion of behaviors manifested by the population of models produced in the experiment. In this way, it is possible to determine under what circumstances there was robustness, that is, when many models exhibited similar behavior for a wide range of hyperparameter combinations and when there was specificity or specialization, when only a few models were able to achieve that result. In short, it provides a broader understanding of how and where tuning happens.

The following profiles have been generated in total: detection profiles, which calculate the percentage of models that presented abnormality detection in a considered cycle and produce such a profile for each type of layer. Then, the lifetime prediction error profile is

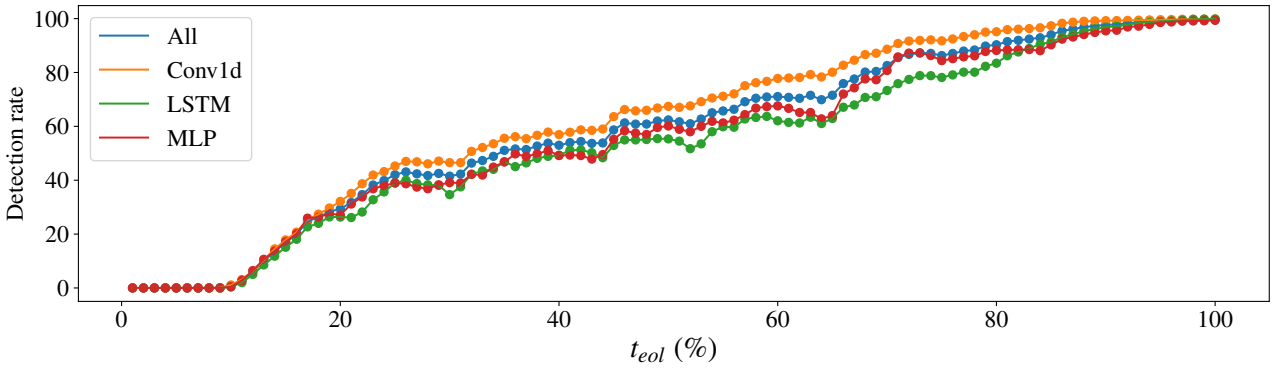
analyzed together with the one that deals with the prediction success rates, which is the ratio in the percentage of models that produced RUL predictions in a given cycle.

As each model is applied in units that differ in life expectancy, it was necessary to adapt the representations to the same reference, the percentage of t_{eol}, which stratifies a scale of normalized cycles, similar to the one used in Section 4.1. However, there is an increase in the procedure regarding the filling of gaps. It is understood that to assign a match to the normalized cycles, the actual cycles are re-scaled based on t_{eol} and then passed through a decimal lease. As a consequence, some interspersed gaps with no addressed quantity may appear. In that case, mitigating actions are taken. For the detection, success rate, and sensor profiles, simple interpolations were used and for the forecast errors, the maximum rule was used, choosing, from two consecutive samples, the one with the highest module and its sign was preserved.

The detection profiles in Figure 24 reveal a gradual evolution of the detection rate for each layer type. At least some model manages to perceive abnormalities at about 15 cycles, below that, no incongruities can be noticed by any of the tested models. The mean time that marks the transition to the transformed degraded state for normalized cycles is 21. Predictions made before this instant are classified as false positives. It is conclusive that the fraction of abnormality indications erroneously made in NOC data is equivalent to 39.65% of the predictions. By comparing the progression between detectability curves, it is inferred that Conv1d types had a much more prominent evolution than LSTM's and MLP's, which in turn manifested very similar detection behaviors. That is, by ablation of the hyperparameters in the search space, the answer was similar. The convolutional models were the only ones that shared the formation of a plateau of 98 to 100% detection rate above 85 cycles, indicating that it is a robust behavior for Conv1d's to notice the abnormality in the remaining 10% normalized cycles of the asset's life. And finally, the degradation in the half-life of the units (50%) was noted by about 60% of all detection models, so the arbitrary selection of a configuration combination within the searched space is probabilistically favorable to the event.

The prognostic profiles emphasize a common trend of convergence of the tested models with the approach of the end of life of the asset. This demonstrates that even with a random choice of a combination of configurations, within the searched space, there is a great chance that the forecast result will be usable, but not optimized, especially in the final quarter of the units' life. These profiles are arranged between Figures 25a and 25b and are complemented by Figure 25c, which prints the outliers, which were not added in the previous figures in order to not complicate the visualization.

Figure 24: Detection profiles for Conv1d, LSTM, MLP and all models, in comparizon.



Source: Author

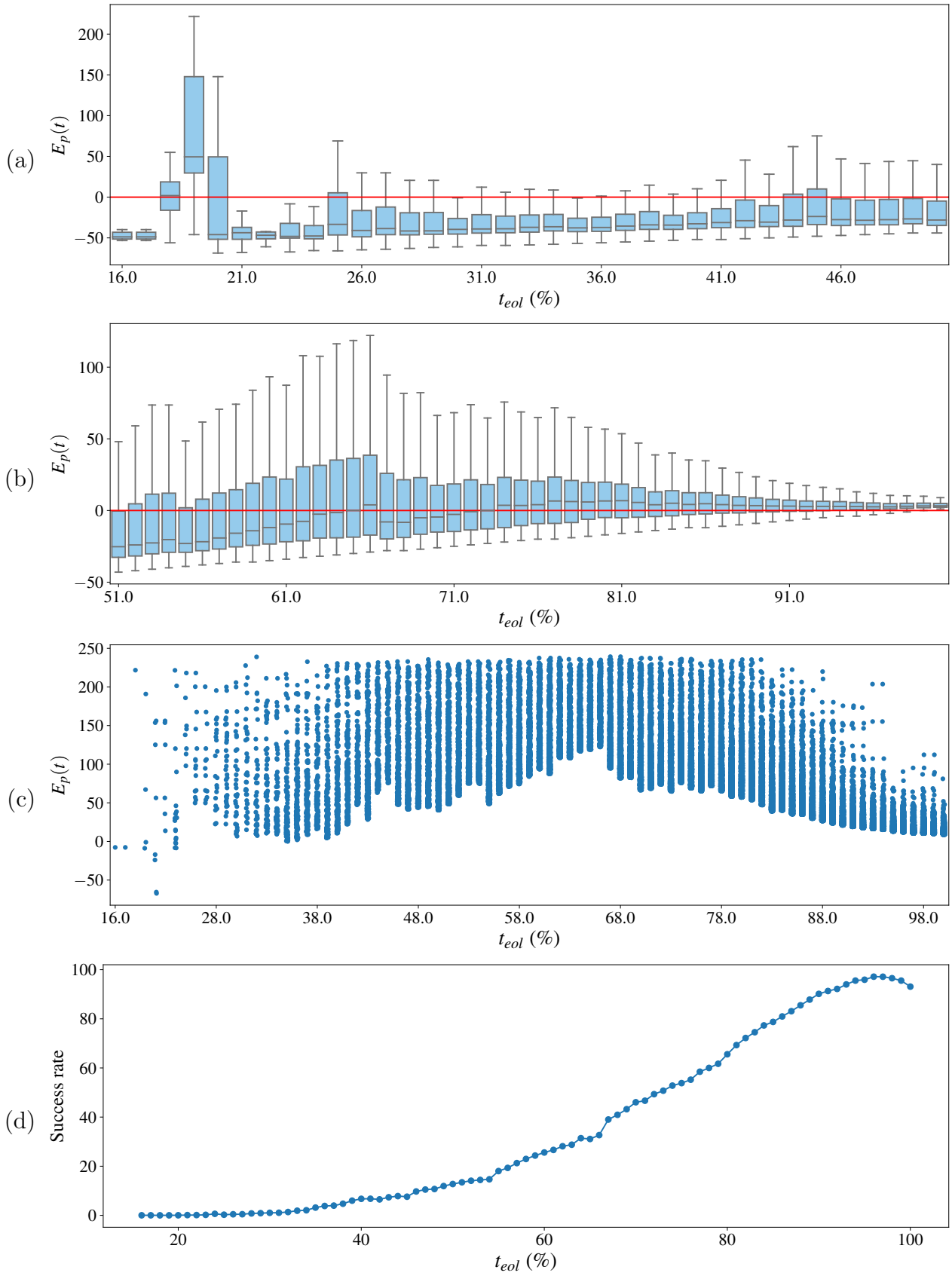
Furthermore, the expression of robustness and specificity of the progression of errors can be interpreted with the aid of Figure 25d, which informs about the advance of prediction rates, that is, the percentage of prediction models that carried out RUL projections in a given normalized cycle. At the beginning of life, little spurization and also a low frequency of predictions are consolidated, as a result of low detectability. There is a jump in error between cycles 18 and 20, which is credited as an isolated event due to the low prediction success rate. This, when contrasted with the detection rate, demonstrates a less pronounced curve. For example, in the half-life below 20 % of all estimation attempts were effectively carried out. This is because the decision procedure within the prognostic algorithm is more rigorous and requires the presence of a noticeable degradation pattern, therefore, prognostic models are less robust and more sensitive to tuning.

Observing the temporal progression of the errors in Figure 25, it can be noticed that the models are susceptible to making underestimations of the RUL up to the half-life, precisely from 55 to 60 cycles. After that, they change their behavior and begin to overestimate. A prediction that anticipates the t_{eol} would, in theory, be a more conservative and therefore a better option. The reason for the occurrence of this transition is the change in the angular coefficient of the reconstruction error curves, which usually begin to converge in this period. Before the half-life, degradation patterns tend to be more horizontal, which can generate edge instabilities as the curve is being fitted, such as the Runge phenomenon in polynomial interpolation. The errors reduced as the degradation pattern became more evident, as well as the skewness of the distributions and there was an approximation of the median to zero. Thus, there is a tendency to reach, disregarding the adjacent outliers, a prediction horizon of 20 average in 91 cycles. Furthermore, there was robustness for the search space for errors less than 12 in the last 5 operation cycles.

The small drop in the success rate at the end of the normalized life could be caused by the instruction used for interpolation.

Outliers persisted over almost the entire progression of useful units, extending from the upper limit of the boxplot to the maximum error limit tolerated by the prediction algorithm. They are not so frequent at the beginning of the asset's life, as there are few error samples in this region, which end up defining the entire prediction error variability. In the last quarter of life, outliers tend to follow the tendency for the prognosis error to converge to zero, as well as the median. There are two plausible explanations for the existence and distribution of the spurization of the models' prediction errors in Figure 25c. One is the complete degeneration of a model that has received a bad setup combination. The other would be isolated peaks of errors resulting from punctual nonconformities in the $r(t) \times t$ curve, which in turn may reflect spikes of reconstruction errors or interferences in the decision function within the prognostic algorithm. The hypothesis of isolated error peaks in the temporal progression of predictions of a single model is the one that would most affect the reliability of the prediction task as a whole because the faulty model could not be discarded immediately. Further investigations would be necessary to determine which of the hypotheses is the preponderant one, although it is highly probable that both coexist in the experiment carried out. It is possible to affirm, then, that there is a plateau or dispersion of outliers characteristic of the estimates made in the middle of the assets life cycle. Its effect can be understood as follows: by selecting a model with an arbitrary combination of settings, there is a high probability that it contains at least one outlier.

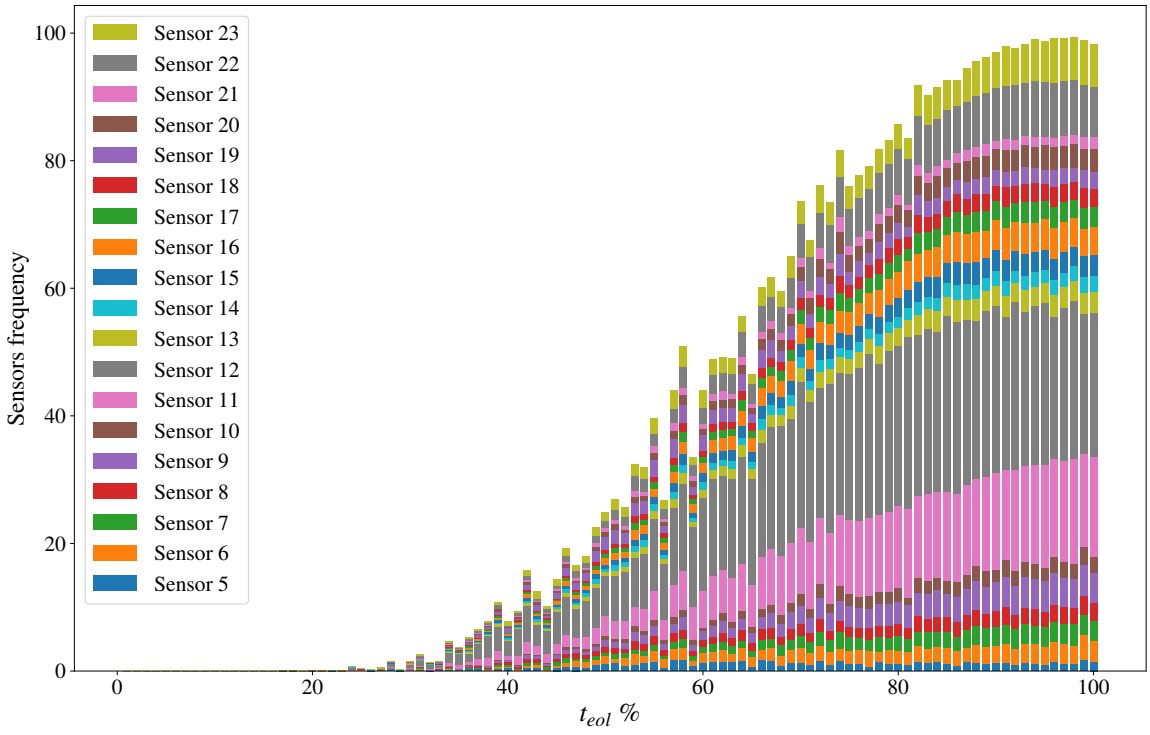
Figure 25: Prediction error profile for the (a) first half of the assets life and (b) second half. In (c) Outliers beyond the box-plot whiskers and (d) prognosis success rate over time.



Source: Author

The sensor profile in Figure 26 enables a better understanding of the decision-making process that takes place within the Algorithm 1. It exposes the frequency with which the projection of degradation contained in a sensor was used to calculate the RUL in a given normalized cycle, considering as a totality (100%) all the models that were successful in making a prediction in this cycle. It is evident that at least two of the 18 sensors used were highlighted in the prognostic task. These are sensors 11 and 12, which correspond to the respective magnitudes of total temperature at HPT outlet ($T48$) and total temperature at LPT outlet ($T50$) in accordance to Appendix A. Both have a closer response to changing modifiers, that are internal variables of the CMAPSS simulator in which the artificial pattern of deterioration is inserted. The conclusion deduced from this graph is that most of the tested models base their predictions on information from the two sensors mentioned and that, according to the tuning conditions, the configured models can choose to add more sensors in their decision-making, which is a specific behavior. Further investigations are needed to find out whether specimens that based their projections on a more diverse array of sensors also demonstrated an improvement in performance indicators.

Figure 26: Sensors time profile. For each cycle displays, in a accumulated form, the frequencies $f_i = s_i/N_{RTF}$, where N_{RTF} is the total number of RTF trajectories which reach the cycle. s_i are the number of times that the sensor i was chosen by the decision rule for all the prognostic models. Sensor 23 is a virtual channel that represents the mean of the REs for all the real sensors.



Source: Author

4.1.2.3 Correlation analysis, clustering of metrics, and model ranking

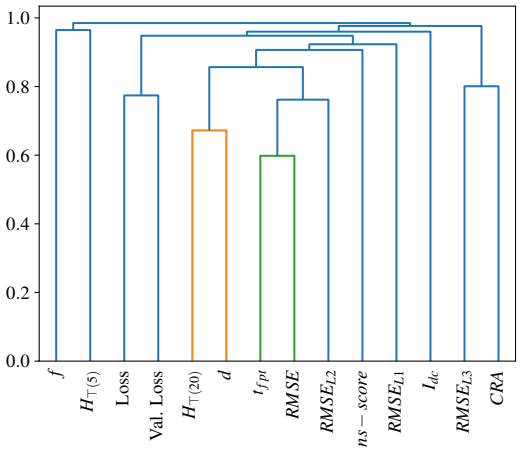
The analysis of the interaction of the metric space elements with each other is of equal importance to that of the metric space with the hyperparameters, discussed in Section 4.1.2.1. It allows checking the consistency of the choice of the set of metrics and also the identification of redundancies. As the metric space is wide and measures different requirements for detection and prognosis models, it is necessary to compress it into a more succinct group of objective functions that serve as support for ordering the most appropriate elements for the fulfillment of the required functionalities, according to the criteria assigned by the user.

The clustering step is divided into two: the theoretical determination of a grouping or grouping proposal for the metrics according to their respective functionalities. And the second would be a comparison of this proposal against an empirical cluster derived from data from tests conducted with the formulated models. The experimental part of the aggregation was carried out with the aid of the dendrogram in Figure 27, which used the expression below as a measure of the distance between each pair of parameters:

$$cm_{ij} = \sqrt{1 - corr_{ij}^2} \tag{4.1}$$

where $corr_{ij}$ is the Pearson correlation measure between i and j metrics and cm_{ij} is the correlation matrix entry resulting from the calculation. The above manipulation is necessary so that the correlation-based distance can satisfy the triangular inequality, a requirement for it to be mathematically formalized.

Figure 27: Dendogram representating the hyerarquical clustering of the performance metrics according with the distance in Equation 4.1.



Source: Author

In theory, performance metrics can be classified into those that quantify the accuracy of RUL estimates, those that assess the extent and consistency of RUL predictions over time, and those related to detection excellence. Table 9 groups the metrics into the suggested theoretical cluster. In this table, detectability evaluates the capacity to perform abnormality detection given a pre-established set of requirements and extensibility measures whether a condition is satisfied in a range of the assets life or the length of the range itself. Prediction error is self-explanatory.

Table 9: Theoretical aggregation of metrics for model ranking. Weights are applied in the MCDA weighted sum method. Each one is attributed in proportion to the relative importance of the variable inside the cluster. Such understanding have been built upon the outcomes of the previous sections.

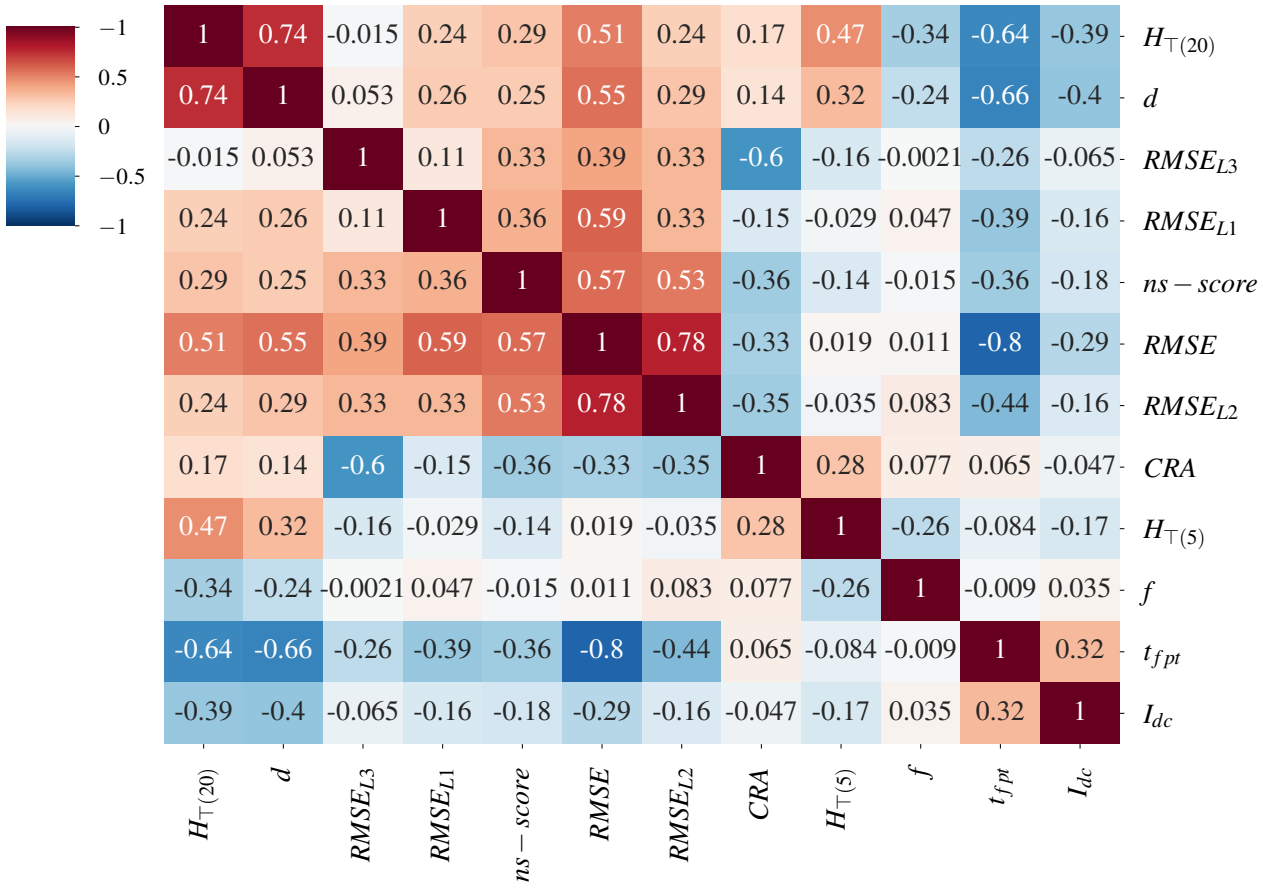
	Description	Hyperparemeter	Weight
Objective 1	Prediction Error	$RMSE$	0.1
		$RMSE_{L3}$	0.5
		$RMSE_{L2}$	0.3
		$RMSE_{L1}$	0.1
Objective 2	Extensibility	t_{fpt}	0.5
		$H_{\top(5)}$	0.2
		$H_{\top(20)}$	0.3
Objective 3	Detectability	I_{dc}	0.1
		d	0.7
		f	0.2

The empirical cluster, in turn, provides an aggregation that separates (in two large groups) the error metrics, detection, extensibility, and neural network adjustment. The closeness of the values of cost and validation cost of the detection metrics is understandable since the convergence of networks primarily affects this task. The proximity of detectability and extensibility measures can be explained by the fact that the earlier the onset of the degradation pattern is perceived, the greater the time available until t_{eol} for predictions to be made. And in the same line of reasoning, prognosticators who achieved good convergence results, that is, with greater prognostic horizons, were those who began to estimate the half-life of the units in proximity.

Two nonconformities stand out in the dendrogram: one is the proximity of the t_{fpt} to the total $RMSE$ and the other is the aggregation of the discontinuity index with the measurements of prediction errors. The first stems from a strong inverse correlation of -0.8 between the metrics, as indicated in Figure 28. The reason is simple: when t_{fpt} is greater, the first forecasts tend to have a high error value and when it is smaller, only

the good estimates, that is, closer to the t_{eol} and with a smaller error are remaining for the calculation of the $RMSE$ total. From the Pearson correlation entries to t_{fpt} shown in Figure 28, it is evident that this is an independent metric that should at least be considered as an eliminatory requirement for selecting sets of configurations by tuning.

Figure 28: Pearson's correlation coefficient between the performance metrics.



Source: Author

The question of the I_{dc} having assumed such a position in the dendrogram may also be due to the low correlation with the other metrics. The discontinuity index is affected by instabilities in the transition zone between NOC and the degraded state, but it did not seem to have a decisive effect on detectability or prognosis. This implies that models with high I_{dc} and high false-positive rates caused by one-off runs can still map well almost the entire extent of the degradation pattern.

The ranking of models was carried out with the aid of a simple multicriteria decision technique (MCDA) based on a weighted sum method (SAATY; GASS, 1954). Each of the three major metric clusters in Table 9 was transformed into three objective functions, two of which make up eliminatory criteria, by assigning weights to the metrics according

to the same table. Weights are normalized to 1 and metric values are scaled by maximum and minimum values, fulfilling the requirements of this MCDA technique. Prior to the ranking process, models are filtered by the following eliminatory requirements, determined according with the interpretation of the results in Sections 4.1.1 and 4.1.2.1:

1. t_{fpt} below 70
2. d above 0.85
3. $H_{\top(5)}$ above 5
4. $H_{\top(20)}$ above 20
5. $RMSE_{L1}$ below 40

The general ranking is quantified from the product $objf1 * objf2, objf1$ concerns the minimization of the error in the estimation of the RUL, while $objf2$ concerns the extensibility. Maximizing both implies selecting the models that made the most predictions, most anticipated to t_{eol}, and with the smallest errors. The top ten according to this criterion are listed in Table 10, while the values of the objectives function are presented in Table 11. In the latter, Euclidean distance in the criterion space is defined as $E = \sqrt{(Objective_1)^2 + (Objective_2)^2 + (Objective_3)^2}$ and $Product = Objective_1 * Objective_2$. For all the columns higher is better.

Table 10: Ten best models designated by the decision criteria applied together with the hyperparameters that characterize them.

ID	Type	Timesteps	Neurons	Kernels	Overlap	Moving
1	Conv1d	500	(16, 8)	(15, 10)	0	1000
2	Conv1d	700	(16, 4)	(10, 5)	50	1000
3	Conv1d	700	(16, 8)	(15, 10)	30	1000
4	Conv1d	300	(8, 4)	(25, 10)	20	2000
5	Conv1d	500	(8, 4)	(25, 10)	50	2000
6	Conv1d	500	(16, 8)	(10, 10)	50	2000
7	MLP	400	(64, 16)		50	2000
8	Conv1d	700	(16, 4)	(25, 10)	30	1000
9	Conv1d	200	(32, 16)	(10, 5)	50	2000
10	Conv1d	500	(16, 4)	(25, 10)	50	2000

Among the best models, there is a preponderance of convolutional types, as expected and according to the analysis carried out in Section 4.1.2.1, on the response of the metric space to hyperparameters. Except for 1, all had an overlapping of at least 20 and

smoothing by a moving average of at least 1000. The high moving average value confirms the importance of using post-processing to control spurization and its impact on the excellence of the models. Temporal steps, neurons, and kernels were the hyperparameters with the greatest diversity in Table 10, showing that there is a wide range of different combinations for these that can lead to optimality. Figure 29 allows, through a graph of parallel categories, a better understanding of the relationship between the hyperparameters and the objective functions. In it, continuous variables are discretized in ten equally sized subintervals (categories). The size of the bars and fractions of the bars are proportional to the number of the samples in the category. Color scale is defined by the product (Table 10), for which higher is better.

Table 11: Objectives functions values evaluated for the models in Table 10.

ID	Objective 1	Objective 2	Objective 3	Euclidean Distance	Product
1	0.828082	0.763157	0.985545	1.496472	0.631956
2	0.818694	0.742548	0.442243	1.190469	0.607920
3	0.815029	0.704773	1.000000	1.470026	0.574410
4	0.829056	0.692286	0.950494	1.438761	0.573944
5	0.831562	0.670187	0.977980	1.448134	0.557302
6	0.849489	0.643064	0.979010	1.446936	0.546276
7	0.810595	0.664054	0.499849	1.160983	0.538279
8	0.735002	0.714257	0.971218	1.411969	0.524980
9	0.868452	0.591419	0.828715	1.338191	0.513620
10	0.683934	0.733566	0.986230	1.406604	0.501711

Figure 29 elucidates that most of the accepted models are of the Conv1d type, so there is a proportionally greater incidence of it in the best-ranking positions with a high Product value. There was therefore a greater disqualification of models with MLP and LSTM layer types, which did not meet the eliminatory requirements. It is also noticed that a larger subsequence size (700) favored combinations with convolutional layers, while intermediate sizes contributed to MLP ones.

Regarding the number of neurons, intermediate amounts of (16,4) and (16,8), with respective compression ratios of 4 and 2, helped Conv1d models. Others with significant presence were (8,4) and (32, 16). However, relationships that concern the other neuron clusters are inconclusive due to their low frequency, considering that the neuron search space is customized for each layer.

In general, models responded better to smaller kernel sizes in the second layer. The tuple that contained the largest size in the first layer (25,10) was compatible with convolutional models with a higher compression rate, as well those with a smaller number

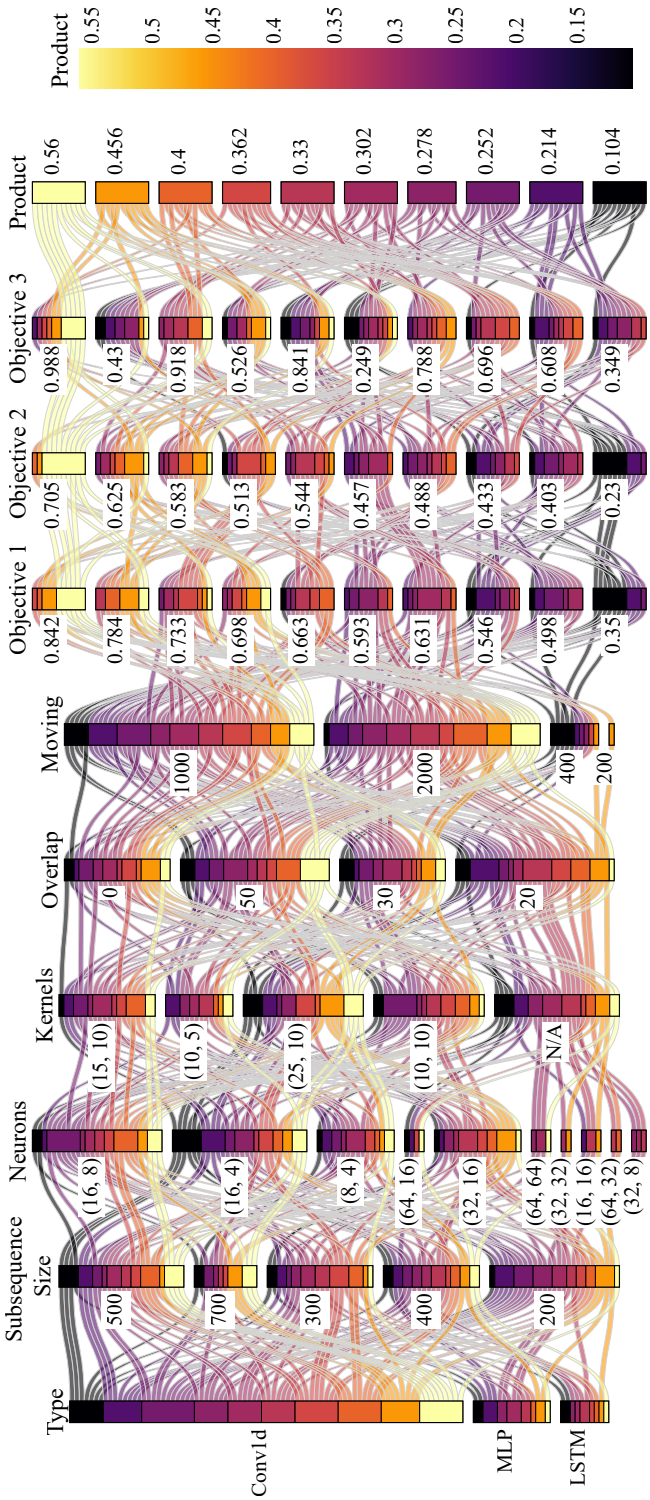
of neurons. Exemplars with large kernels were also those that improved performance in the presence of higher overlapping (50%). Overall, an overlapping rate of 50% benefited more models compared to other percentages.

Concerning to moving averages, higher values confirm the importance of smoothing, already mentioned in the previous sections, for maximizing model performance. It can be seen from Figure 29 that a large number of combinations with small moving averages did not meet the requirements and were disqualified.

Although directly proportional, the relationship between the *Product* and objective functions contains some particularities resulting from the adopted weighting. $Objective_1$ had a strong correlation with *Product*, as well as models with a high $Objective_2$ (>0.705) also exhibited a high *Product*. This premise is also observed for $Objective_2$, but at a slightly lower frequency (75%). However, it is not so evident for the third objective function, where only (50%) of models with high $Objective_3$ showed a high *Product*. Thus, it is recommended to review the weighting of $Objective_3$, if the intention is to promote models with lower prediction errors in the ranking.

In summary, Figure 29 allows for composing chains of relations between hyperparameters. For example, a template type [Conv1d; 700; (16,8) ; (25,10); 50; 2000], in figure label order, is a good candidate, as is [Conv1d; 500; (16, 8); (15.10); 0; 100]. So, it appears that is possible to graphically represent the relationships between hyperparameters through graphics of parallel categories and, therefore, visualize the result of tuning.

Figure 29: Parallel categories diagram show the relationship between the hyperparameters and the objectives functions.



Source: Author

4.2 Limitations, further improvements and future opportunities

The detection and prognosis framework explored in this dissertation was tested in a database manufactured from a simulator of commercial turbofan modular air propulsion systems. The synthetic RTF's derived from this simulator do not encompass all the nuances of a real operating condition of an equipment or engineering system, such that a deeper investigation into the behavior of this set in punctured bases of industrial monitoring systems should be conducted in the future. It was noted that the formulation of such integrated prediction tools consists of shortening two types of fundamental gaps: correspondence to reality and generalization. They elucidate to what extent a CBM system formulated from a given technique meets adherence, usability, and reliability requirements in the predictions consistent with the health status of an asset in its practical context of use. And the second deals with the ability of such a system, or framework, to be exported to different application contexts or the same context in different scales of conditions without compromising its performance. Thus, it measures adaptability in terms of gross equivalent performance capacity and transition effort, that is, the amount and cost of modifications for a system to function in this other context.

The diversification of tests for the real bases and the production of dedicated software based on a more detailed specification and oriented to the final user - maintenance engineer(s) and industrial process operators - can be a way to reduce the gaps and allow for closer experimentation with effective CA, although it requires partnerships with companies. In general, in practical scenarios machines are subject to intermittent operating conditions, interspersed with failures and outages, and discontinuities caused by changes in process or product specifications or improvements. This state of transitivity leads the implemented solutions to degenerate in performance over time if no mitigating action is taken.

In the case of a framework based on deep learning, this implies correcting the obsolescence curve with the setup and frequent training of new models. However, such a solution brings complications for the user, because the requirement for settings and customization is directly proportional to the demand for training in using the program. Another factor to be taken into account is the time dedicated to the deployment of the models, that is, the execution of the software's internal routines and the time for setup or use of the user interface. In general, a factory work routine tends to be fast, and designated employees may not have as much time available to dedicate themselves in front of the interface.

Recommendations to mitigate such effects are the scaling and preparation of dedicated hardware infrastructure, the automation of decision and predefinition of model configuration instructions, and the creation of software functionality that allows the management of processes in progress, either by executing processes in parallel or by hierarchization through a priority list for the execution of routines.

There are also limitations inherent to the use of DL, arising from the demand for large volumes of data and computational resources. The application in the production of methods that employ it requires (on the part of IT departments) the preparation of dedicated infrastructure and training in good machine learning (ML) practices. Furthermore, there are the intrinsic limitations of the constructive characteristics of DL techniques and derived architectures, which are labeled as black boxes. These models are based on the assumption that there is a statistical correspondence between the sample distributions in the training and use domains and their reliability is based on stimulus-response tests, as a resource to avoid vulnerabilities, such as adversary attacks.

Therefore, explainability should be explored in the future, especially if black box models are used for deliberations on critical assets, as they may, in omission circumstances, provide a false impression of security. Because of this, the recommendation for the application of DL in decision systems is that there is a high reward in terms of using the model and a low or tolerable penalty for the generation of erroneous answers. The existence of redundancies and verification mechanisms is also recommended. Unfortunately, risk assessment methodologies for decisions taken with the aid of DL are lacking, according to a survey conducted by the author.

With regard to the grid search experiment, it is possible to expand the search space to variables for which response sensitivity cannot be obtained. It is also indicated to resort to previous procedures, such as univariate ablation and references in the literature, to define this space in order to avoid expenses. The use of an exhaustive strategy is not advised for tuning DL models and was carried out in the context of this work with the purpose of experimentation since there are already search algorithms for more efficient discrete spaces. Some specific suggestions about the developed framework and code are provided in the section below.

4.2.1 Suggestions about the framework and code developed

The code execution time bottlenecks that can be improved are the training of neural networks, using parallel processing in GPU, and the curve fitting routine in Algorithm 1, currently performed by a solver imported by a standard scientific computing library. It was identified that there is a slower return of the function in case of optimization divergence – that is, unfitting the candidate curve – because the optimizer persists for many iterations. Assigning a maximum iteration ceiling slightly improves the problem, however, there is still space for more gains, as these solvers are not natively prepared to deal with dynamic curve fitting conditions and to declassify functions with very bad fitting already in the initial stages. An intervention to be tried lies in the vertical transfer – time domain – and horizontal – the domain of the set of curves – of initial condition parameters for objective functions.

Another point of improvement of the prediction algorithm concerns predictions at the beginning of the degradation process, in which different curves tend to show high R^2, overestimating the remaining lifetime value. This behavior has been already expected, as predictions tend to improve in accuracy as more information about the condition becomes available. However, the decision functions must be fixed to avoid outliers that exceed t_{eol} values by more than 300 % or produce atypical variability of predictions between two consecutive moments. One way to do this is by noting that $dRUL_{real}/dt = -1$ to force downgrade outliers. In addition, it is also necessary to better calibrate the decision functions so that the other indicators are taken into account in the hierarchy of functions and sensors, which currently relies heavily on R^2 values.

As for the data pre-processing and reconstruction error post-processing routines, it is feasible to explore new trend-smoothing techniques that do not suppress many samples, which was the case with the moving average used. The application of a moving average of $n = 2000$ points implies losing the equivalent of 30 cycles at the beginning of the abnormality state and contributing to the effect mentioned in the above paragraph. This is a problem for units that had low t_{eol} or that operated for a few cycles in an abnormal condition before failure, such as Unit 14, as there are few samples designated for the prognostic step. Another alternative would be to merge the labeled inputs with those in normal conditions at an offset of at most n before the detection point.

Potential advances in terms of the quantification of uncertainties at different stages of the framework can be made either by using Bayesian neural networks or generative models or by adopting probabilistic regressors for the extrapolation of reconstruction errors.

Other points that aggregate uncertainties are in the attribution of the moment of transition to the degraded state, in the attribution of the prognostic trend limit, and, therefore, in the estimation of the RUL, the latter of which absorbs all the variability arising from the decision-making process within the algorithm of prognosis, as well as inherits the epistemological and random remnants of the models and data sets, respectively. On the other hand, the quantification of multiple sources of uncertainty has a considerable impact on performance, especially if Monte Carlo random sampling techniques are prevalent. The problem of integrating multiple sources of uncertainty within this framework to produce predictive results with safety margins and fulfill specificities of the current regulations in a computationally efficient manner is still open to the author. In the absence of the possibility of doing this globally, it is recommended to identify the factors that most contribute to the variability of the prognosis result, such as the definition of the error limit for prognosis, which has a great impact on performance, as observed in Section 4.1.

4.3 Contributions of this work

The main contribution of this work is a framework for detection and prognosis that uses deep-learning techniques. The framework is capable of delivering a deterministic prognostic result through the recognition of degradation patterns with monotonic growth implicit in a set of monitored variables and updating its prediction when new data is available.

The advantage of this approach is the capacity to be easily implemented in an industrial context, which has the particularity of having an abundance of engineering systems data in NOC and with few recorded faults. Furthermore, real scenarios had lower quality of data labels or unlabeled data. The framework is designed considering this observation since there is no need to previously attribute labels or even discriminate sets of abnormal samples. Another point is that a complete framework that embraces detection and prognostic models worrying about scalability is elaborated. At the end, the proposed framework is more suitable for use in different industrial domains and has an extensive application range because it does not demand physical information or intensive knowledge about the fault's nature and its signature in the sensors' readings.

Secondary contributions comprises a metrical assessment procedure to validate the models' performance, which implies also researching a set of metrics designated to evaluate the different requirements necessary for the detection and prognosis models. Thus, some of those metrics are proposed by the author: detection coverage, false-positive coverage

and discontinuity index.

Moreover, an intensive and systematical analysis of the hyperparameters and their influence on the output generated by the framework has been conducted. In that area, this work also innovates with novel ways to visualize performance for outputs derived from large collections of hyperparameters combinations. Specifically, through the use of time profile plots.

By the end, this work contributes to understanding the current limitations in the application of similar techniques in industrial scenarios, together with concerns about scalability. A flowchart that provides instructions for the implementation of a software module on a production scale is presented in Appendix C.

5 CONCLUSION AND FUTURE WORKS

A method centered on a framework for detection and prognosis that employs autoencoder deep-learning techniques, therefore data-driven, was conceived. This framework was capable of dealing with degradation patterns of monotonic growth implicit in a set of monitored variables from a database derived from a traditional simulator in the literature of the area. The application example originated from it and met the conditions of having a reasonable number of channels, therefore a wide characteristic space, and the presence of transient operational conditions, which are the takeoff and landing of aircraft.

In addition, a set of metrics specifically designed to quantify the functional goals and effectiveness of the models produced was verified with its help. Such development configures a first step for CES implementation in real operating conditions. In fact, the conception of a process flow facilitates the transition to industrial systems by organizing and standardizing internal routines to be translated into software specifications, making it conveniently modular. In effect, standardization provides a basis for iterative development of improvements and for cases of domain transfer, as well as ensuring uniformity of benchmarking.

Nonetheless, the synthetic RTFs derived from this simulator do not encompass all the nuances of a real operating condition of an equipment or engineering system, such that a deeper investigation into the behavior of this framework on the databases of industrial monitoring systems should be conducted in the future. In practical scenarios, machines are subject to changes for a variety of reasons. This state of transitivity leads the implemented solutions to degenerate in performance over time if no action is taken. In the case of a framework based on deep learning, this implies correcting the obsolescence curve with the setup and frequent training of new models, which is time-consuming.

It is worth mentioning the restrictions inherent to the use of DL, caused by the demand for large volumes of data and computational resources, as well as those induced by the constructive aspects of DL techniques and derived architectures. These are black box models, so means of explainability should be researched in the future, especially if

the intention is to employ these models in deliberations on critical assets. Therefore, is recommendable to apply DL only in environments of controlled risks, as risk assessment methodologies for decisions taken with the aid of DL are lacking, according to a survey conducted by the author.

A simple application allows inferring that the autoencoder models developed are capable of equating performance with a baseline one that uses simple linear regression on pre-processed signals. Even though it is not possible to present a model at this stage of development that surpasses the simplest alternative in all metric requirements, it is noteworthy that there is still a large margin of adjustment available through hyperparameter modulations, neural network architectures, post-processing adjustments for reconstruction errors, among others, in order to achieve greater gains.

Then, conducting a grid-search experiment enabled a deeper understanding of the sensitivity of the hyperparameters that comprise the framework and what is their response to the metric space, as well as of the interaction of the metrics with each other. Therefore, the response can be mapped to the vast majority of hyperparameters, as well as distinguishing robust from specializable behaviors, that is, it allows inferring the applicability of tuning.

It was also possible to attest the tuning capacity from a multi-criteria decision method, with theoretical objective clusters, but proven by empirical findings, and thus perform the ranking of the models. In general, it is demonstrated, although with weak evidence, the feasibility of using deep learning for health prognosis and management.

Future work should aim to implement improvements both in the developed code and in the framework. The code execution time could be improved by the adjustment of the curve fitting routine in the prognosis algorithm. Also, by trying to control the prediction error at the beginning of the degradation process, where the estimations of the remaining life tend to be overestimated. In the processing routines of the reconstruction errors, it is recommended to explore new trend-smoothing routines that do not suppress many samples.

Another important point to be investigated is the uncertainty quantification at different stages of the framework, which reflects in the estimation of the RUL being in fact a random variable. It is a challenge to quantify and integrate multiple sources of uncertainty to produce prognostic results that fulfill the specificities of the current regulation in a computationally efficient manner.

5.1 Publications

The following publications have been produced within the scope of the present work:

- ROSA, T. G. da et al. Data driven fault detection in hydroelectric power plants based on deep neural networks. In: LEVA, M. C. et al. (Ed.). *32nd European Safety and Reliability Conference*. Dublin, Ireland: Research Publishing, 2022. p. 8.
- ROSA, T. G. d. et al. Semi-supervised framework with autoencoder-based neural networks for fault prognosis. *Sensors*, v. 22, n. 24, 2022. ISSN 1424-8220. Available in: ⟨https://doi.org/10.3390/s22249738⟩.

REFERENCES

AHMAD, R.; KAMARUDDIN, S. An overview of time-based and condition-based maintenance in industrial application. *Computers and Industrial Engineering*, Elsevier Ltd, v. 63, n. 1, p. 135–149, 2012. ISSN 03608352. Available in: ⟨https://doi.org/10.1016/j.cie.2012.02.002⟩.

AHMED, H. O.; WONG, M. L.; NANDI, A. K. Intelligent condition monitoring method for bearing faults from highly compressed measurements using sparse over-complete features. *Mechanical Systems and Signal Processing*, v. 99, 2018. ISSN 10961216.

BABU, G. S.; ZHAO, P.; LI, X. L. Deep convolutional neural network based regression approach for estimation of remaining useful life. In: . [S.l.: s.n.], 2016. v. 9642. ISSN 16113349.

BARUAH, P.; CHINNAM, R. B. Hmms for diagnostics and prognostics in machining processes. *International Journal of Production Research*, Taylor Francis, v. 43, n. 6, p. 1275–1293, 2005. Available in: ⟨https://doi.org/10.1080/00207540412331327727⟩.

BERGSTRA, J. et al. Algorithms for hyper-parameter optimization. In: . [S.l.: s.n.], 2011.

BROTHERTON, T. et al. A testbed for data fusion for engine diagnostics and prognostics. In: *Proceedings, IEEE Aerospace Conference*. [S.l.: s.n.], 2002. v. 6, p. 6–6.

CHAO, M. A. et al. Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. *Data*, v. 6, n. 1, 2021. ISSN 23065729.

CHAO, M. A. et al. Fusing physics-based and deep learning models for prognostics. *Reliability Engineering and System Safety*, v. 217, 2022. ISSN 09518320.

CHESKY, W. *Side view of turbofan jet engine cross section wireframe isolated on blue background*. 2017. Available in: https://www.istockphoto.com/br/foto/vista-lateral-do-wireframe-de-sePhotograph. Accessed in: May 15, 2023.

ENGEL, S. et al. Prognostics, the real issues involved with predicting life remaining. In: *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*. [S.l.: s.n.], 2000. v. 6, p. 457–469 vol.6.

FINK, O. et al. Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, Pergamon, v. 92, p. 103678, 6 2020. ISSN 0952-1976.

FREDERICK, D. K.; DECASTRO, J. A.; LITT, J. S. *User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)*. 2007. Available in: ⟨http://www.sti.nasa.gov⟩.

GOODFELLOW YOSHUA BENGIO, A. C. I. *Deep Learning*. [S.l.: s.n.], 2016.

GUO, J.; LI, Z.; LI, M. A review on prognostics methods for engineering systems. *IEEE Transactions on Reliability*, IEEE, v. 69, n. 3, p. 1110–1129, 2019.

HENG, A. et al. Intelligent condition-based prediction of machinery reliability. *Mechanical Systems and Signal Processing*, v. 23, n. 5, p. 1600–1614, 2009. ISSN 0888-3270. Available in: ⟨https://doi.org/10.1016/j.ymssp.2008.12.006⟩.

HESS, A.; CALVELLO, G.; FRITH, P. Challenges, issues, and lessons learned chasing the "big p". real predictive prognostics. part 1. In: *2005 IEEE Aerospace Conference.* [S.l.: s.n.], 2005. p. 3610–3619.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 13374-1:2003 Condition monitoring and diagnostics of machines — Data processing, communication and presentation — Part 1: General guidelines*. 2003. Available in: ⟨https://www.iso.org/standard/21832.html⟩.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 13381-1:2015 - Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines*. 2015. Available in: ⟨https://www.iso.org/standard/51436.html⟩.

JANSSENS, O. et al. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, v. 377, 2016. ISSN 10958568.

JARDINE, A. K. S.; LIN, D.; BANJEVIC, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, v. 20, p. 1483–1510, 2006. ISSN 0888-3270.

JAVED, K. et al. Enabling health monitoring approach based on vibration data for accurate prognostics. *IEEE Transactions on Industrial Electronics*, v. 62, 2015. ISSN 02780046.

KATIPAMULA, S.; BRAMBLEY, M. R. Review article: Methods for fault detection, diagnostics, and prognostics for building systems—a review, part i. *HVAC&R Research*, Taylor Francis, v. 11, n. 1, p. 3–25, 2005. Available in: ⟨https://doi.org/10.1080/10789669.2005.10391123⟩.

KONG, Z. et al. Convolution and long short-term memory hybrid deep neural networks for remaining useful life prognostics. *Applied Sciences (Switzerland)*, v. 9, 2019. ISSN 20763417.

LEE, J. et al. Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. *Mechanical Systems and Signal Processing*, v. 42, p. 314–334, 2014. ISSN 0888-3270.

LEI, Y. *Intelligent fault diagnosis and remaining useful life prediction of rotating machinery*. [S.l.: s.n.], 2016. 1-366 p. ISBN 9780128115350.

LEI, Y. et al. Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical Systems and Signal Processing*, v. 104, 2018. ISSN 10961216.

LEWIS, S.; EDWARDS, T. Smart sensors and system health management tools for avionics and mechanical systems. In: *16th DASC. AIAA/IEEE Digital Avionics Systems Conference. Reflections to the Future. Proceedings.* [S.l.: s.n.], 1997. v. 2, p. 8.5–1.

LI, X.; DING, Q.; SUN, J. Q. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, v. 172, 2018. ISSN 09518320.

LIAO, L. Discovering prognostic features using genetic programming in remaining useful life prediction. *IEEE Transactions on Industrial Electronics*, v. 61, 2014. ISSN 02780046.

LIU, S.; FAN, L. An adaptive prediction approach for rolling bearing remaining useful life based on multistage model with three-source variability. *Reliability Engineering and System Safety*, v. 218, 2022. ISSN 09518320.

LIU, X. et al. Fault diagnosis of rotating machinery under noisy environment conditions based on a 1-d convolutional autoencoder and 1-d convolutional neural network. *Sensors 2019, Vol. 19, Page 972*, Multidisciplinary Digital Publishing Institute, v. 19, p. 972, 2 2019. ISSN 1424-8220. Available in: ⟨https://doi.org/10.3390/s19040972⟩.

LU, C. et al. Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification. *Signal Processing*, v. 130, 2017. ISSN 01651684.

LUO, J. et al. Model-based prognostic techniques [maintenance applications]. In: *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.* [S.l.: s.n.], 2003. p. 330–340.

MALHOTRA, P. et al. Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder. 8 2016. Available in: ⟨http://arxiv.org/abs/1608.06154⟩.

MELANI, A. H. de A. et al. A framework to automate fault detection and diagnosis based on moving window principal component analysis and bayesian network. *Reliability Engineering System Safety*, v. 215, p. 107837, 2021. ISSN 0951-8320. Available in: ⟨https://doi.org/10.1016/j.ress.2021.107837⟩.

Montero Jimenez, J. J. et al. Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, Elsevier, v. 56, n. March, p. 539–557, 2020. ISSN 02786125. Available in: ⟨https://doi.org/10.1016/j.jmsy.2020.07.008⟩.

ROBINSON, E. I. *Filtering and uncertainty propagation methods for model-based prognosis*. Tese (Doutorado) — École doctorale Informatique, Télécommunications et Électronique, 2018.

ROSA, T. G. d. et al. Semi-supervised framework with autoencoder-based neural networks for fault prognosis. *Sensors*, v. 22, n. 24, 2022. ISSN 1424-8220. Available in: ⟨https://doi.org/10.3390/s22249738⟩.

ROSA, T. G. da et al. Data driven fault detection in hydroelectric power plants based on deep neural networks. In: LEVA, M. C. et al. (Ed.). *32nd European Safety and Reliability Conference*. Dublin, Ireland: Research Publishing, 2022. p. 8.

SAATY, T.; GASS, S. Parametric objective function (part 1). *Journal of the Operations Research Society of America*, INFORMS, v. 2, n. 3, p. 316–319, 1954.

SAXENA, A. et al. Evaluating prognostics performance for algorithms incorporating uncertainty estimates. In: . [S.l.: s.n.], 2010. ISSN 1095323X.

SAXENA, A. et al. Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and Health Management*, v. 1, 2010. ISSN 21532648.

SAXENA, A. et al. Damage propagation modeling for aircraft engine run-to-failure simulation. In: *2008 International Conference on Prognostics and Health Management*. [S.l.: s.n.], 2008. p. 1–9.

SI, X.-S. et al. Remaining useful life estimation – a review on the statistical data driven approaches. *European Journal of Operational Research*, v. 213, p. 1–14, 2011. ISSN 0377-2217.

SIKORSKA, J. Z.; HODKIEWICZ, M.; MA, L. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, v. 25, p. 1803–1836, 2011. ISSN 08883270.

SMITH, A. E.; COIT, D. W.; LIANG, Y.-C. A neural network approach to condition based maintenance: case study of airport ground transportation vehicles. *IMA Journal of Management Mathematics on Maintenance, Replacement and Reliability*, Citeseer, 2003.

TAO, J.; LIU, Y.; YANG, D. Bearing fault diagnosis based on deep belief network and multisensor information fusion. *Shock and Vibration*, v. 2016, 2016. ISSN 10709622.

THIRUKOVALLURU, R. et al. Generating feature sets for fault diagnosis using denoising stacked auto-encoder. In: . [S.l.: s.n.], 2016.

VACHTSEVANOS, G. et al. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. [S.l.: s.n.], 2007.

WU, W.; HU, J.; ZHANG, J. Prognostics of machine health condition using an improved arima-based prediction method. In: *2007 2nd IEEE Conference on Industrial Electronics and Applications*. [S.l.: s.n.], 2007. p. 1062–1067.

WU, X. et al. A hybrid classification autoencoder for semi-supervised fault diagnosis in rotating machinery. *Mechanical Systems and Signal Processing*, v. 149, 2021. ISSN 10961216.

YAN, W.; YU, L. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In: . [S.l.: s.n.], 2015. ISSN 23250178.

YOON, A. S. et al. Semi-supervised learning with deep generative models for asset failure prediction. 9 2017. Available in: ⟨http://arxiv.org/abs/1709.00845⟩.

ZHANG, B.; ZHANG, L.; XU, J. Degradation feature selection for remaining useful life prediction of rolling element bearings. *Quality and Reliability Engineering International*, v. 32, 2016. ISSN 10991638.

ZHAO, Z. et al. Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study. *ISA Transactions*, v. 107, 2020. ISSN 00190578.

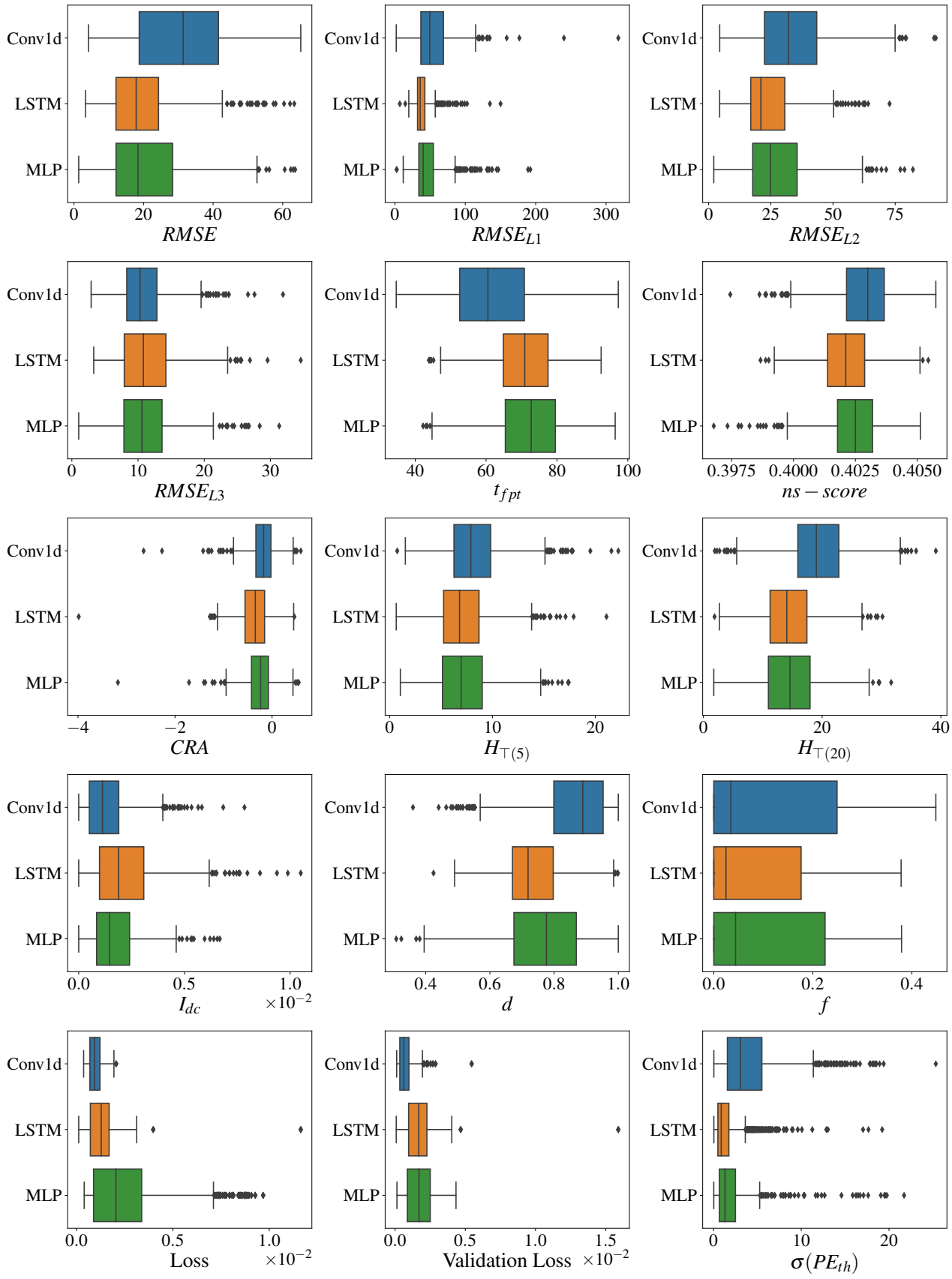
APPENDIX A – LIST OF SENSORS

Table 12: List of indexed sensors variables and their description.

Index	Variable	Description	Units
5	alt	Altitude	ft
6	Mach	Flight Mach number	
7	TRA	Throttle–resolver angle	%
8	T2	Total temperature at fan inlet	°R
9	T24	Total temperature at LPC outlet	°R
10	T30	Total temperature at HPC outlet	°R
11	T48	Total temperature at HPT outlet	°R
12	T50	Total temperature at LPT outlet	°R
13	P15	Total pressure in bypass-duct	psia
14	P2	Total pressure at fan inlet	psia
15	P21	Total pressure at fan outlet	psia
16	P24	Total pressure at LPC outlet	psia
17	Ps30	Static pressure at HPC outlet	psia
18	P40	Total pressure at burner outlet	psia
19	P50	Total pressure at LPT outlet	psia
20	Nf	Physical fan speed	rpm
21	Nc	Physical core speed	rpm
22	Wf	Fuel flow	pps

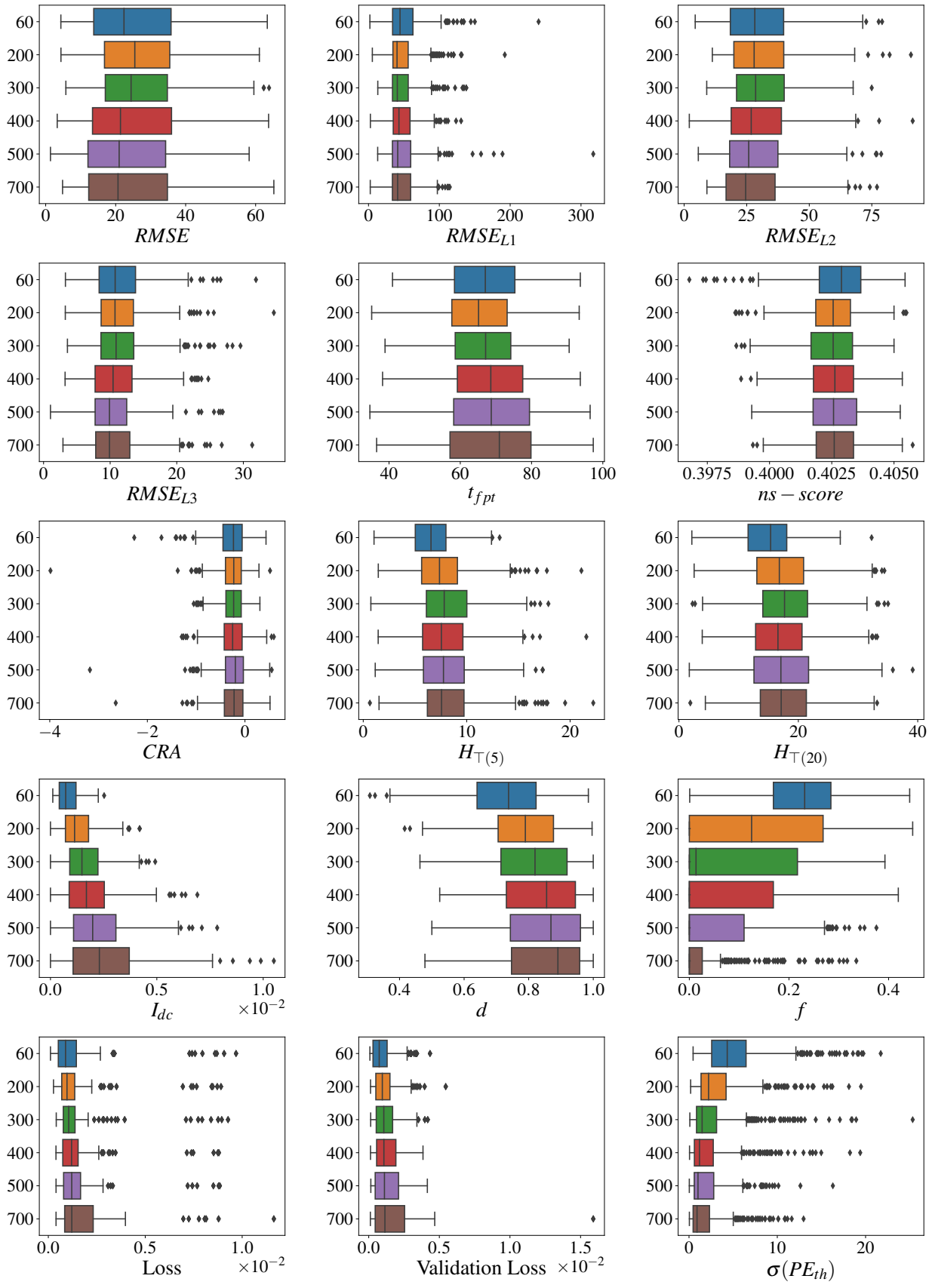
APPENDIX B – COMPLETE METRICAL RESPONSE

Figure 30: Metrical response for types of layers.



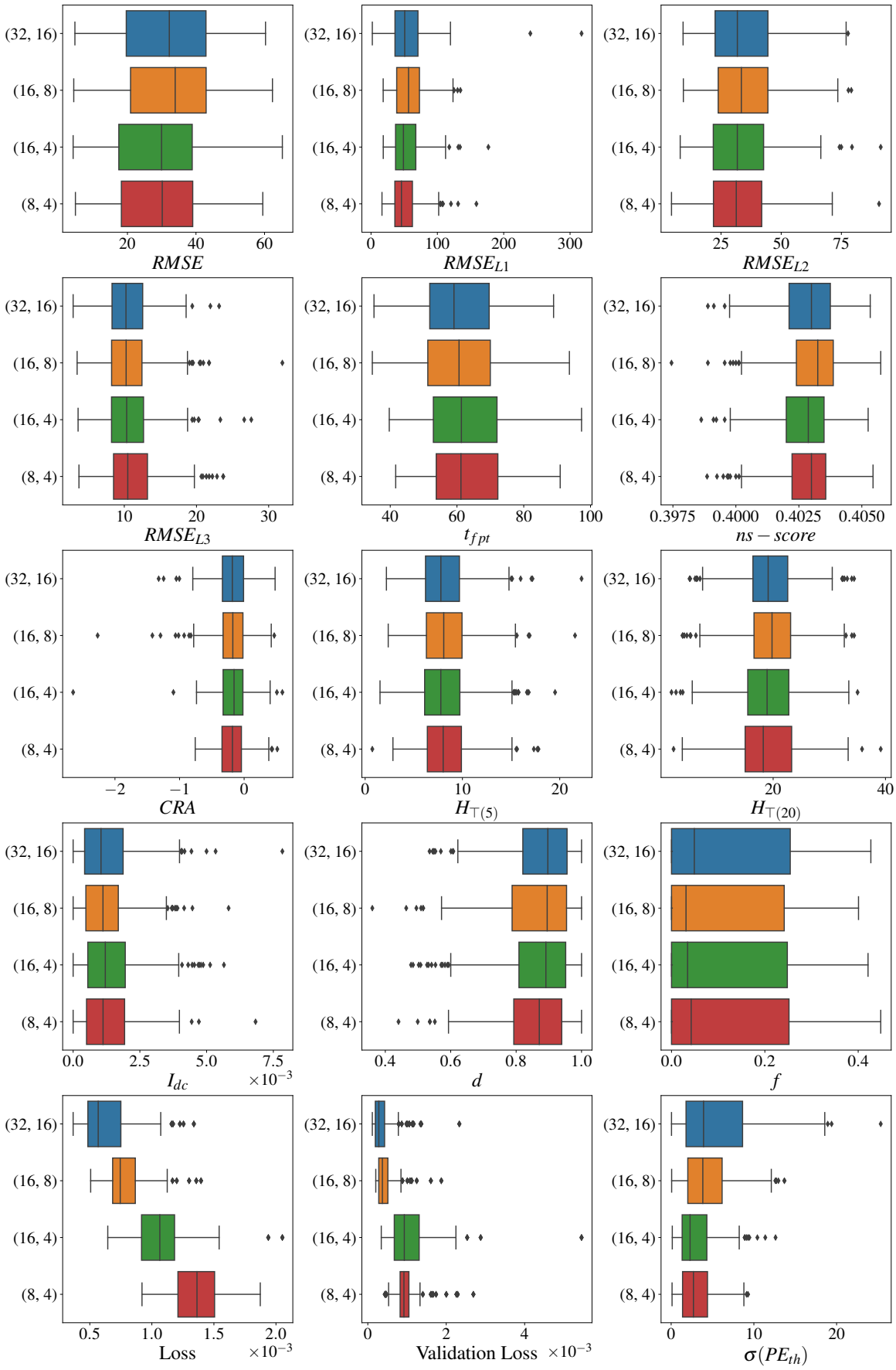
Source: Author

Figure 31: Metrical response for subsequence size.



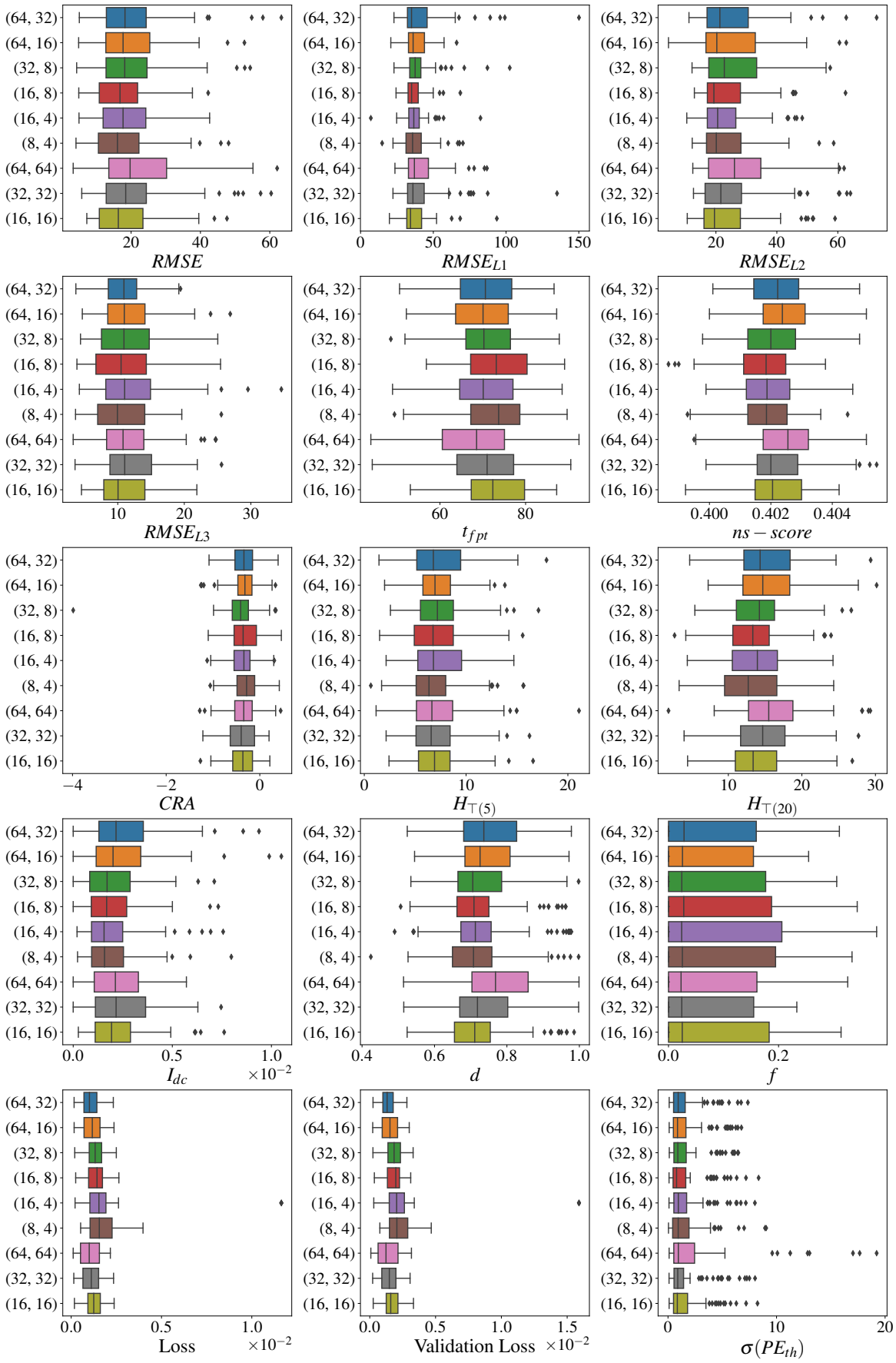
Source: Author

Figure 32: Metrical response for different neurons combinations in Conv1d.



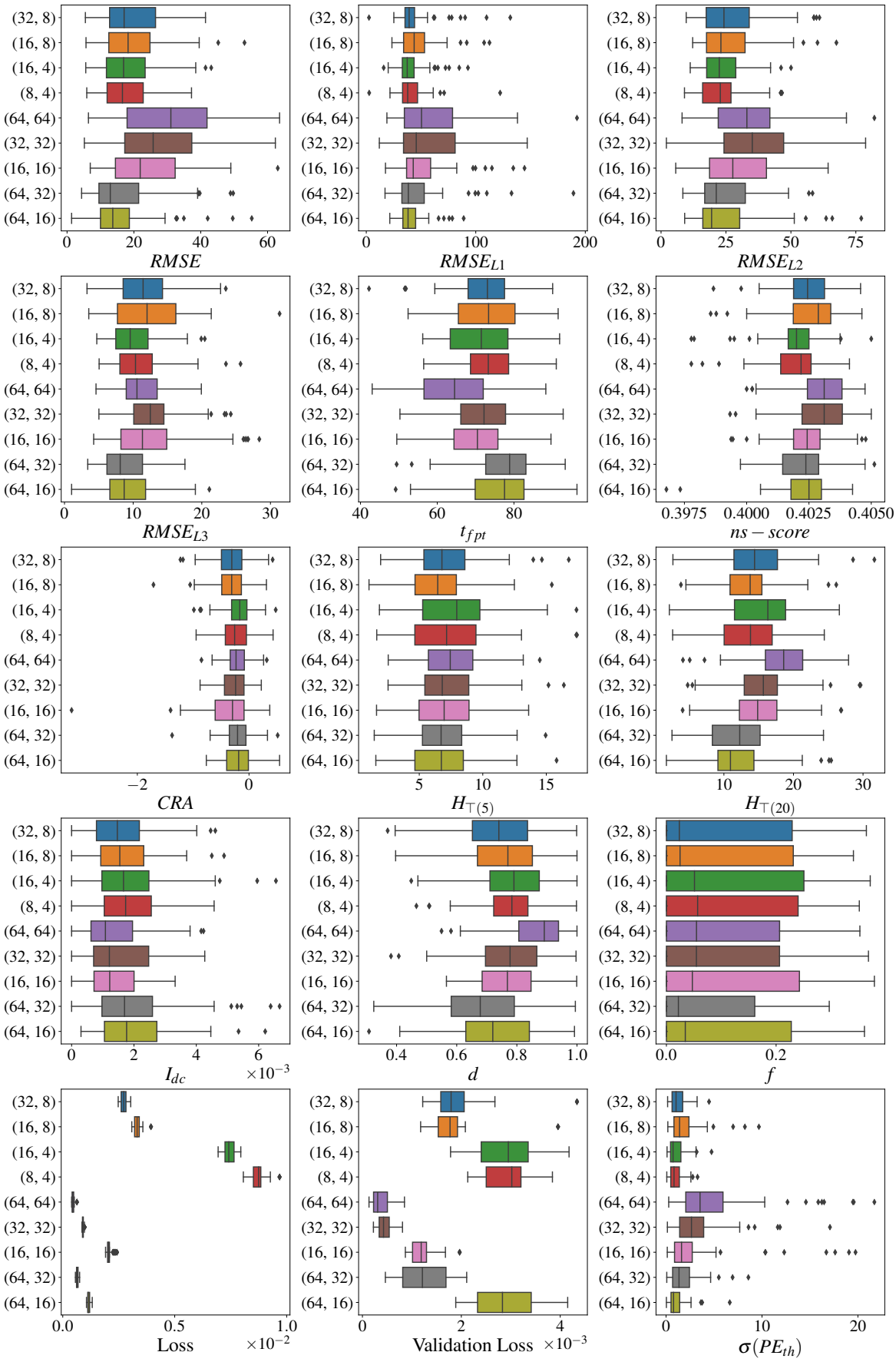
Source: Author

Figure 33: Metrical response for different neurons combinations in LSTM.



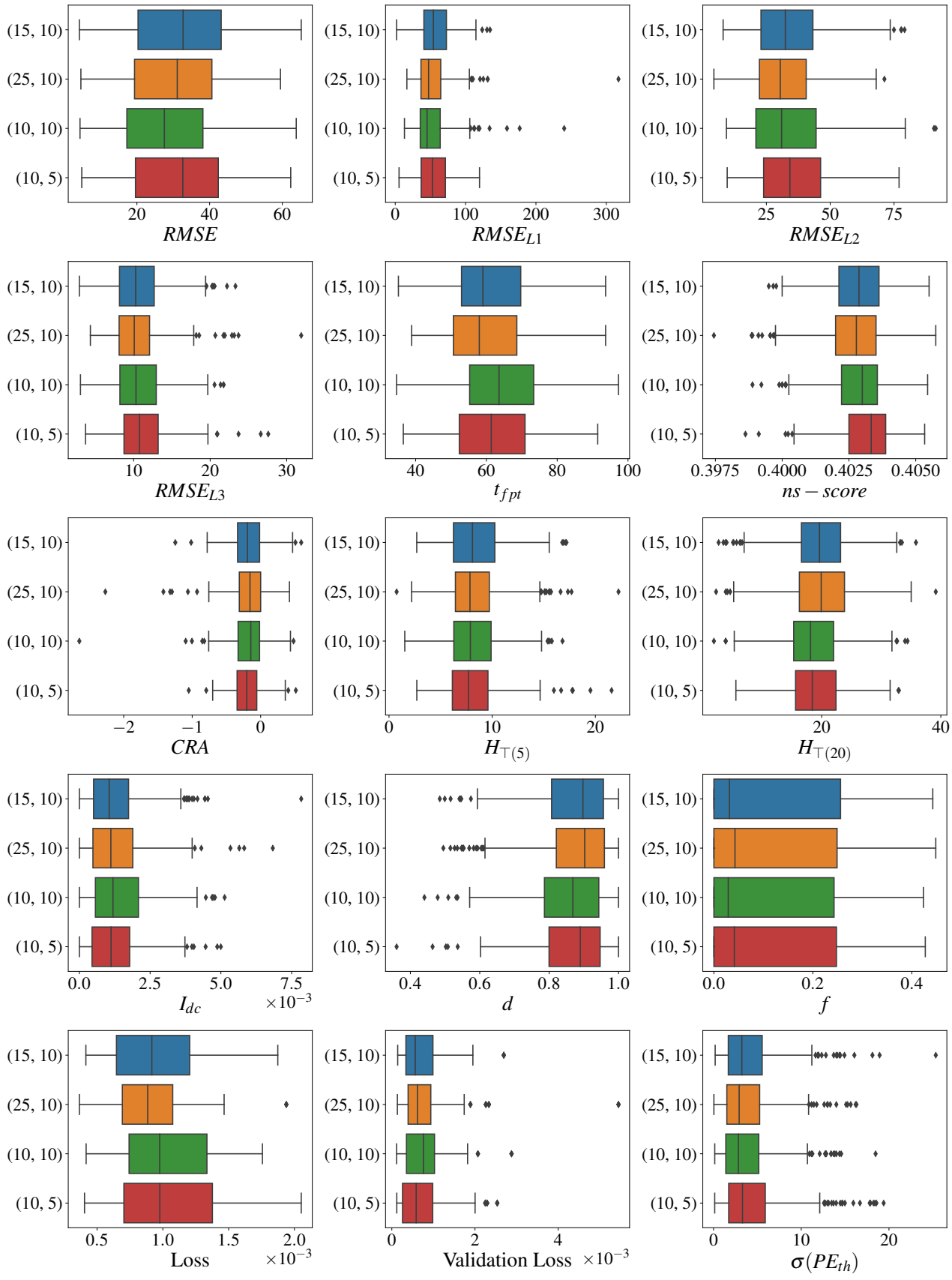
Source: Author

Figure 34: Metrical response for different neurons combinations in MLP.



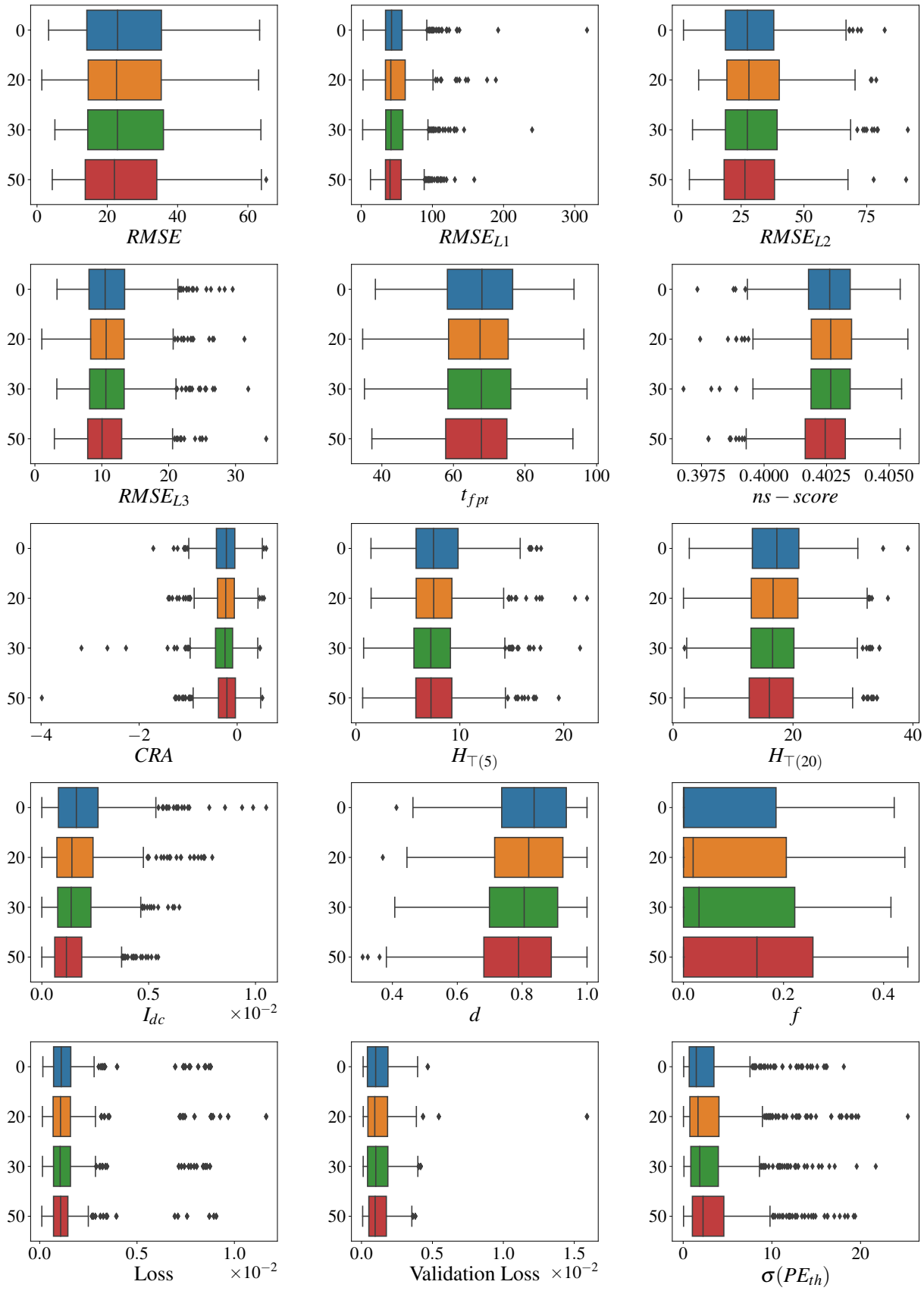
Source: Author

Figure 35: Metrical response for different kernels size combinations.



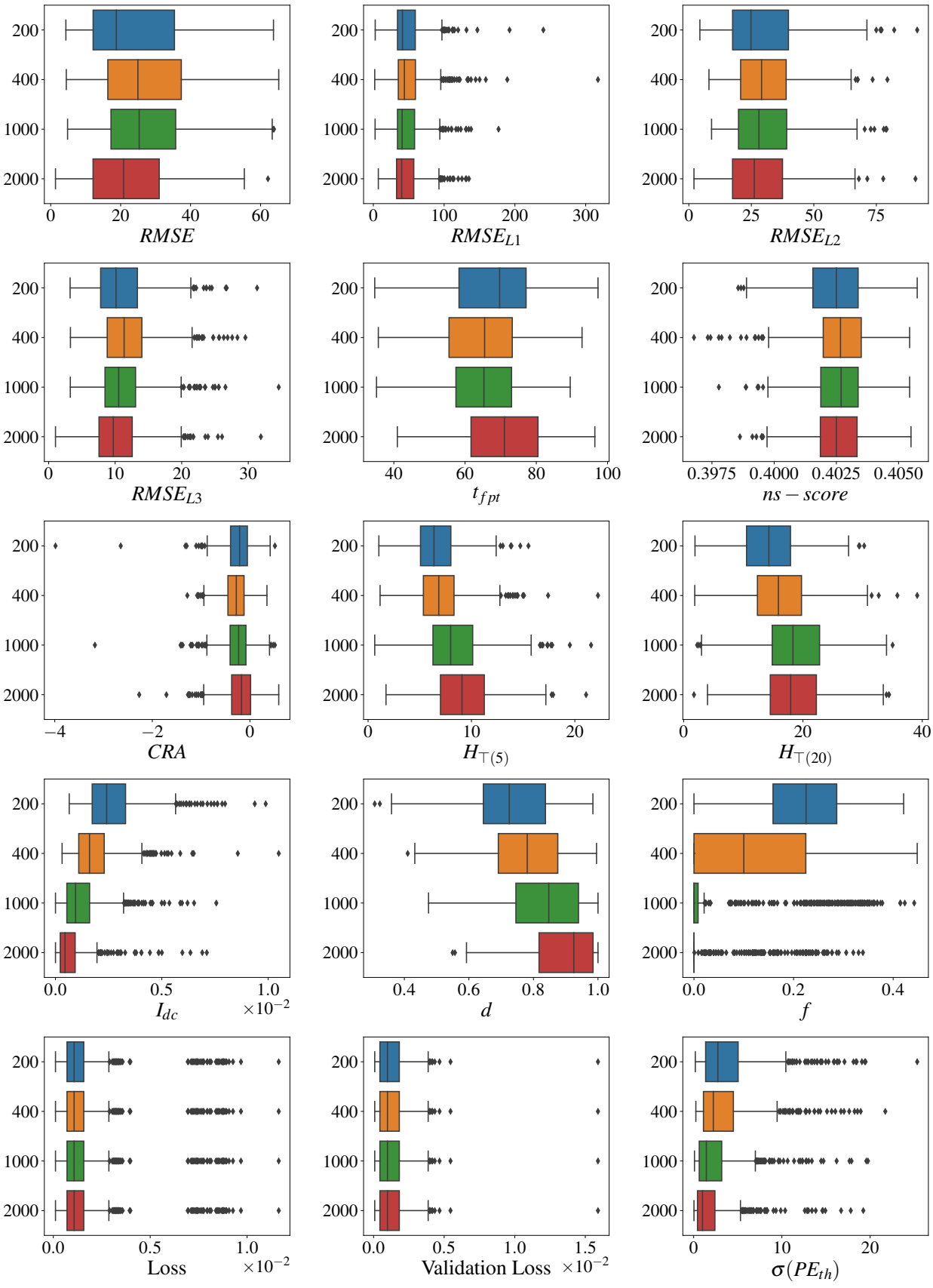
Source: Author

Figure 36: Metrical response for subsequence overlapping.



Source: Author

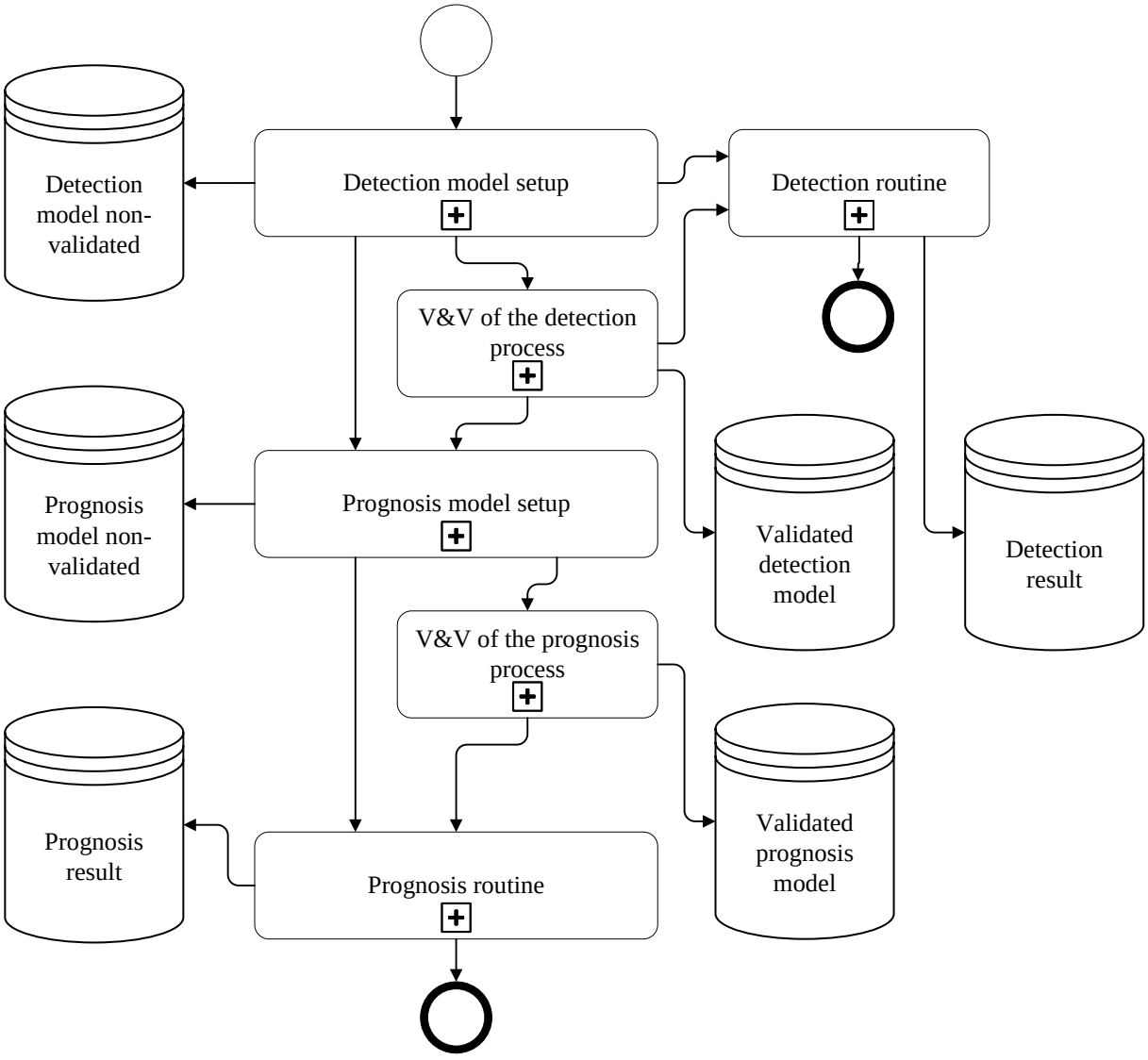
Figure 37: Metrical response for moving average window.



Source: Author

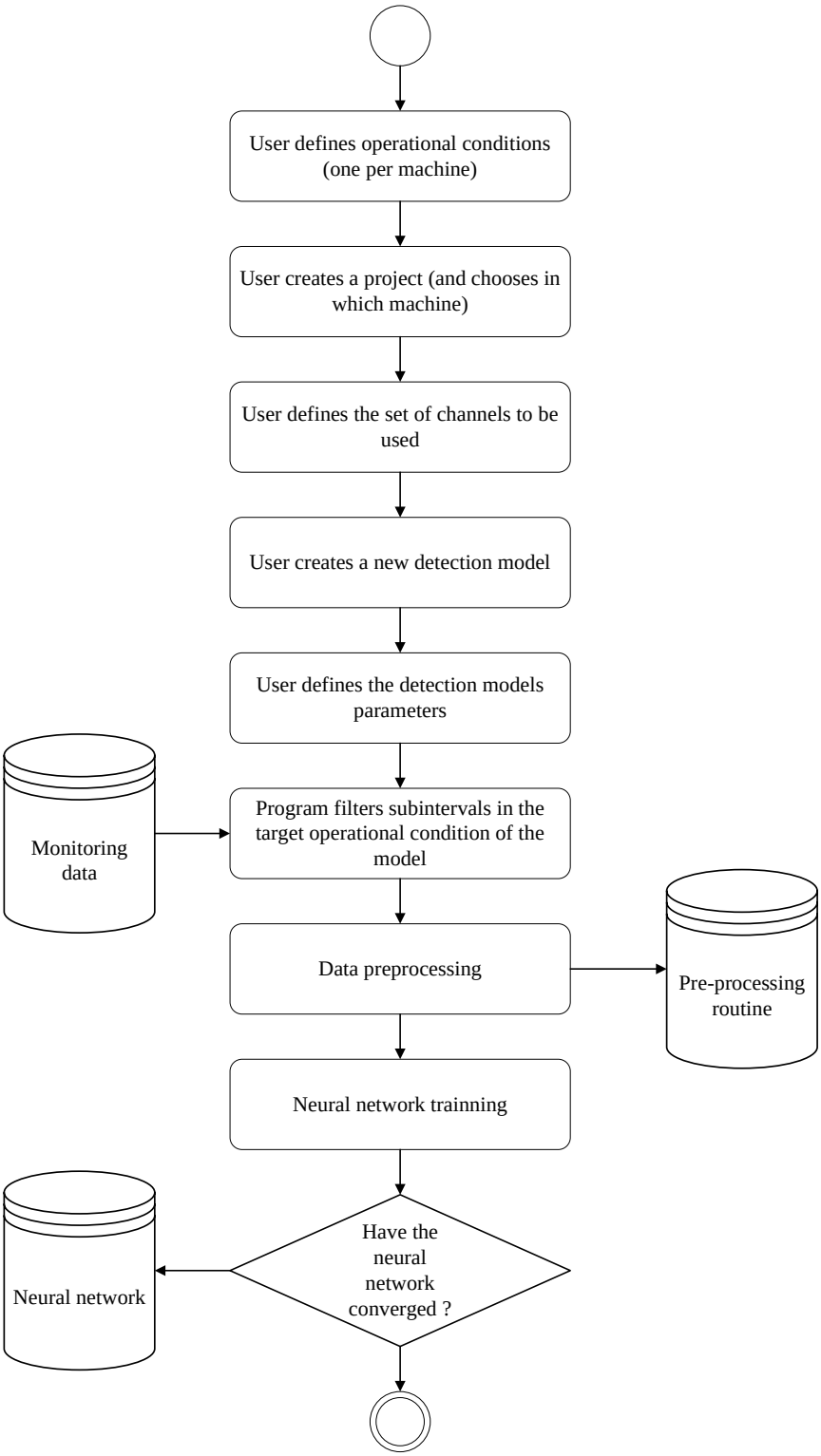
APPENDIX C – FLOWCHARTS

Figure 38: Overview of the detection and prognosis process.



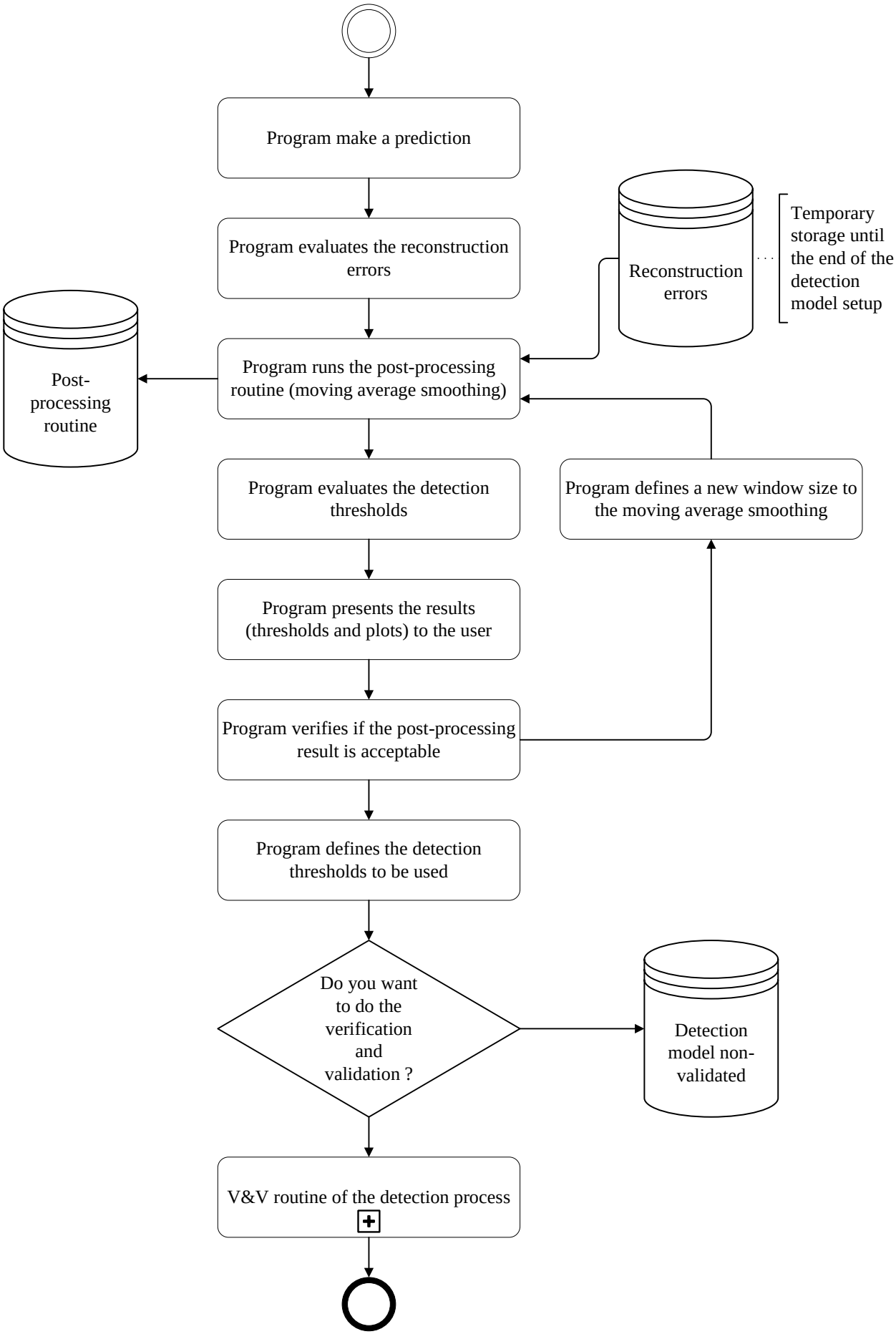
Source: Author

Figure 39: Detection model setup. Part 1.



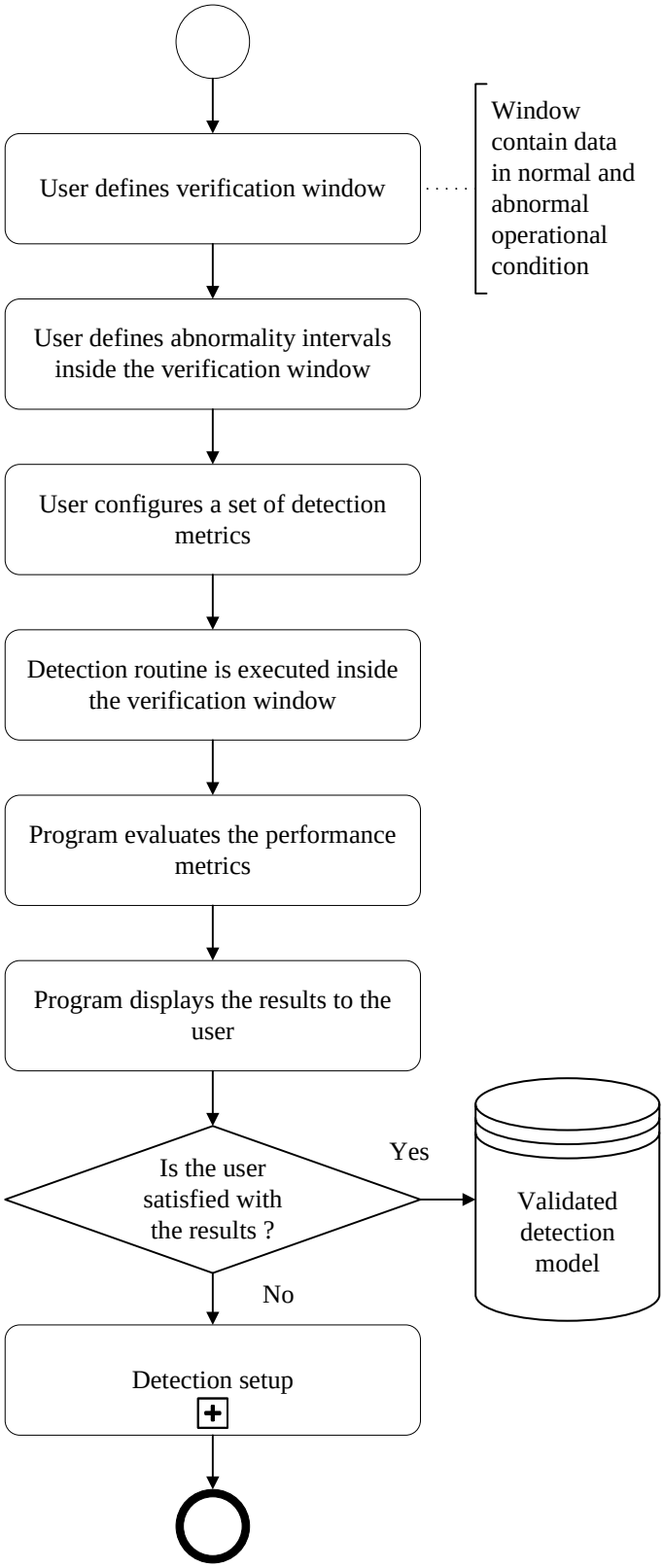
Source: Author

Figure 40: Detection model setup. Part 2.



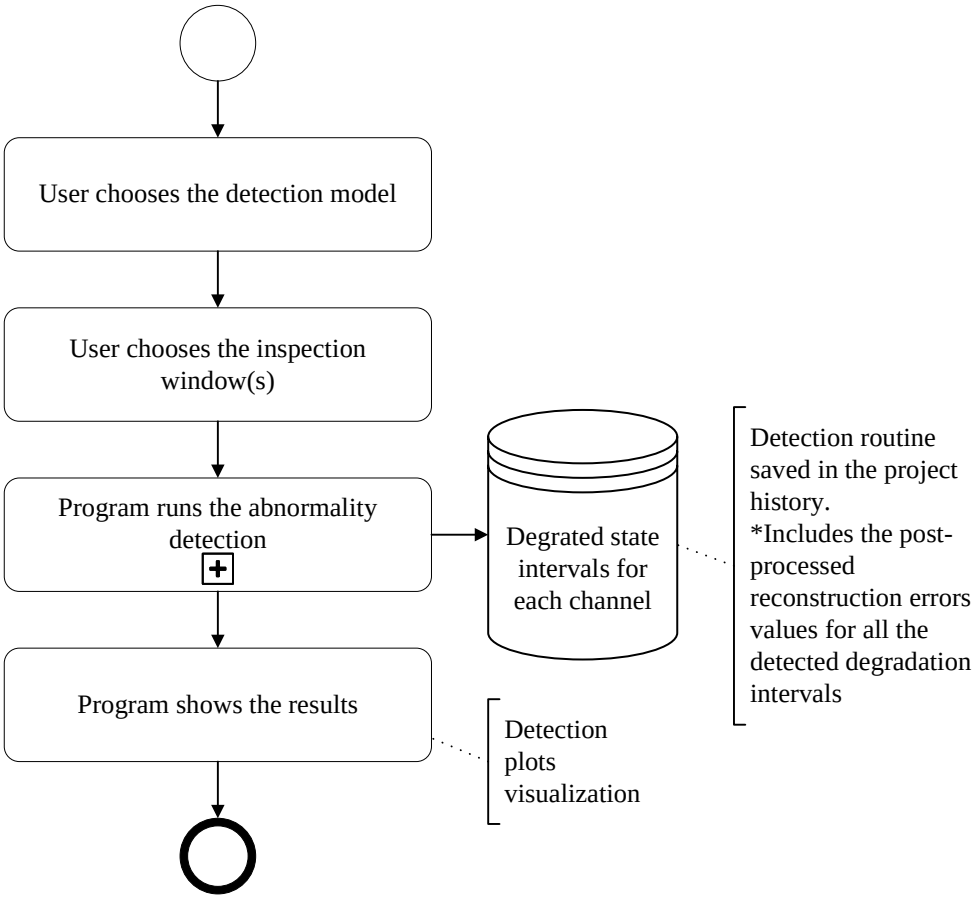
Source: Author

Figure 41: Verification and validation of the detection process.



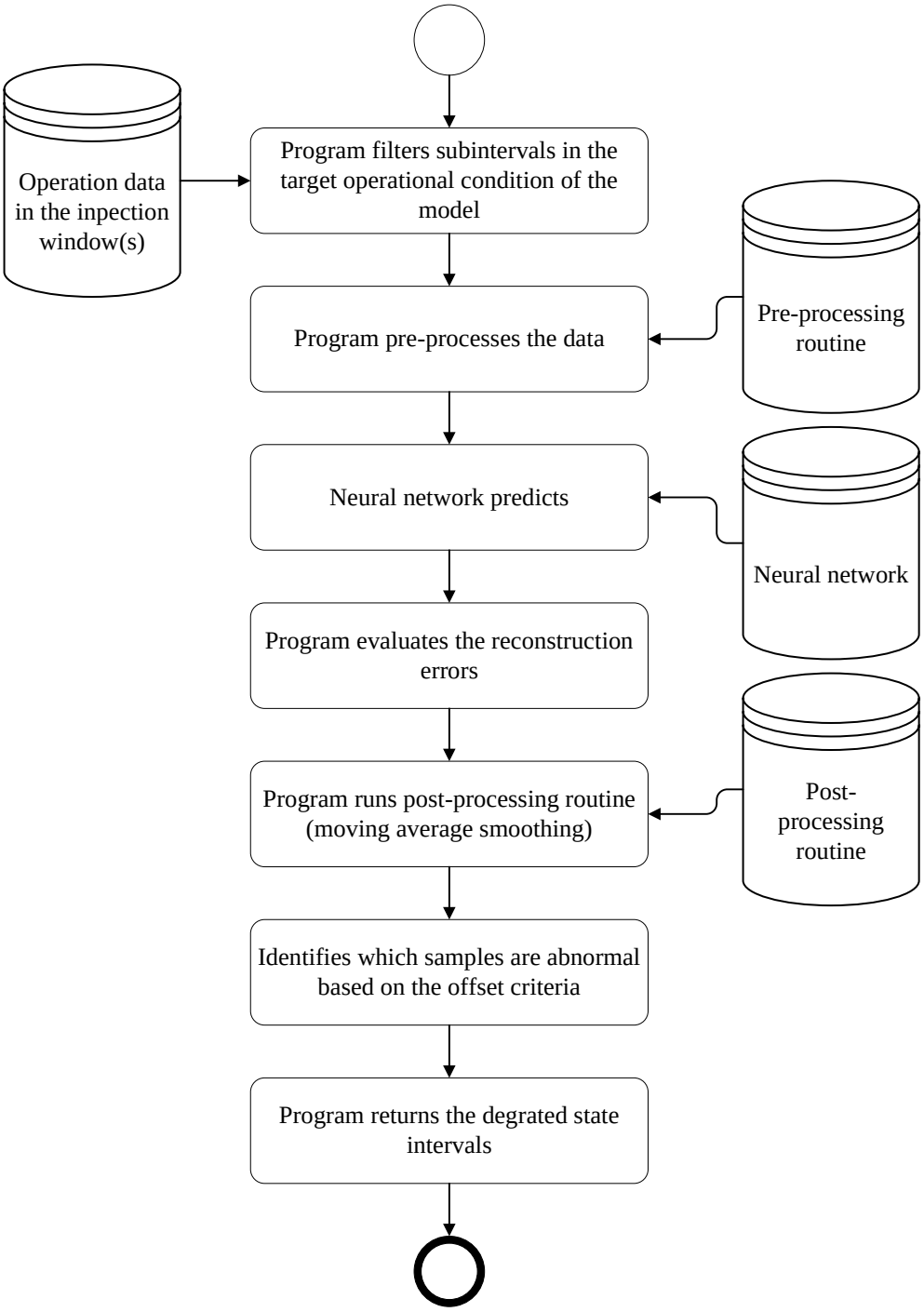
Source: Author

Figure 42: Detection routine.



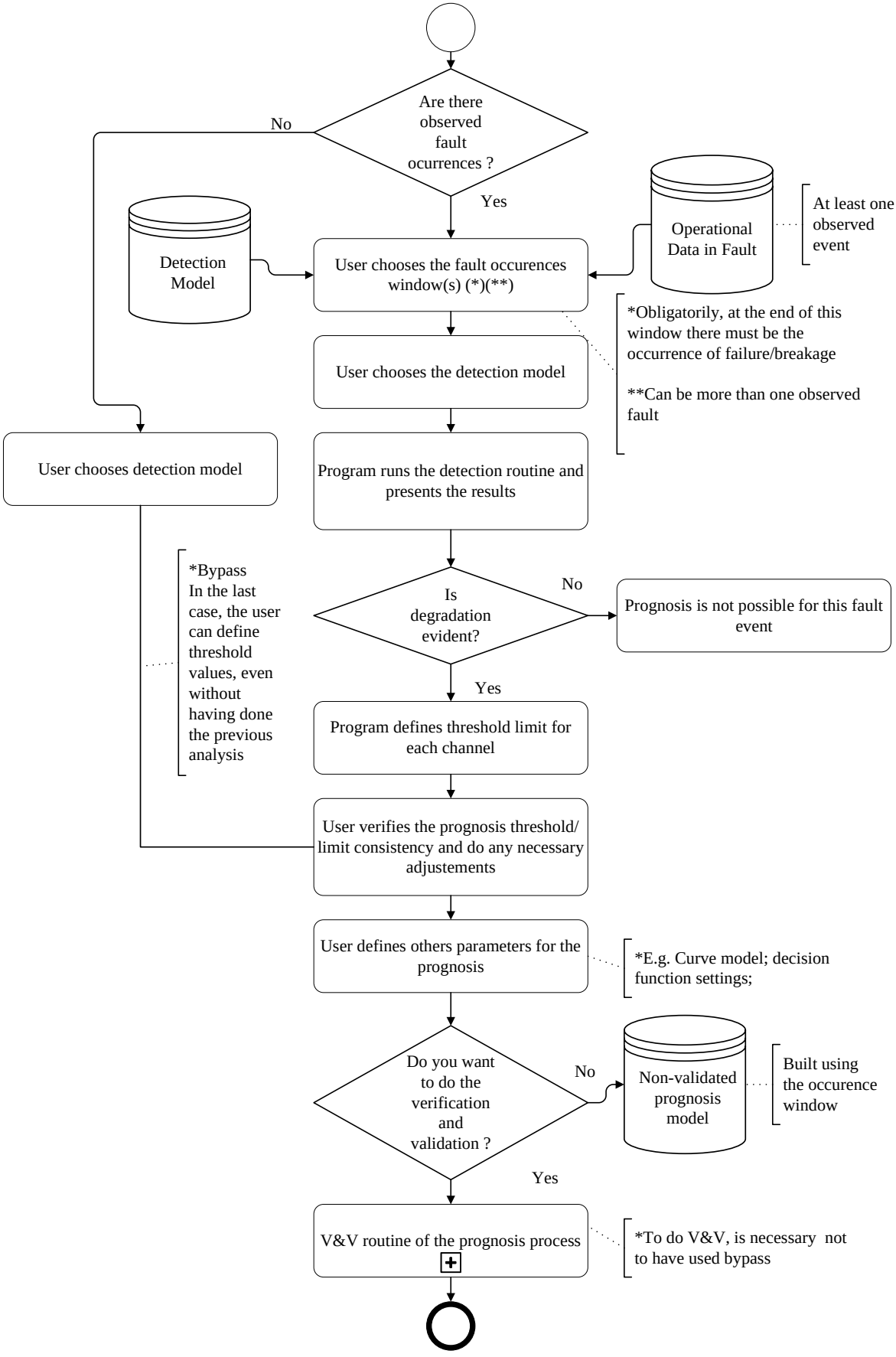
Source: Author

Figure 43: Abnormality detection procedure executed by the program.



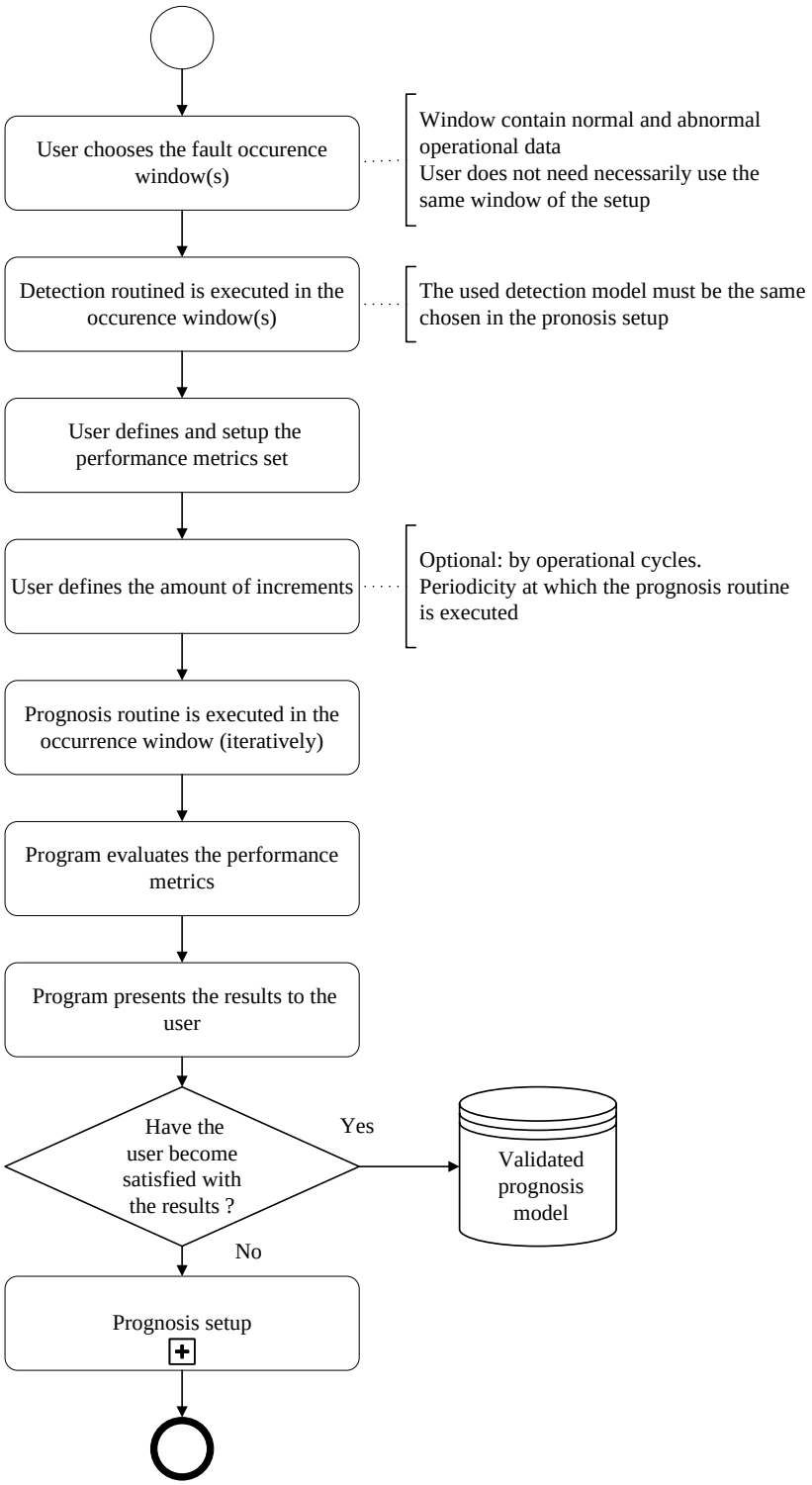
Source: Author

Figure 44: Prognosis model setup.



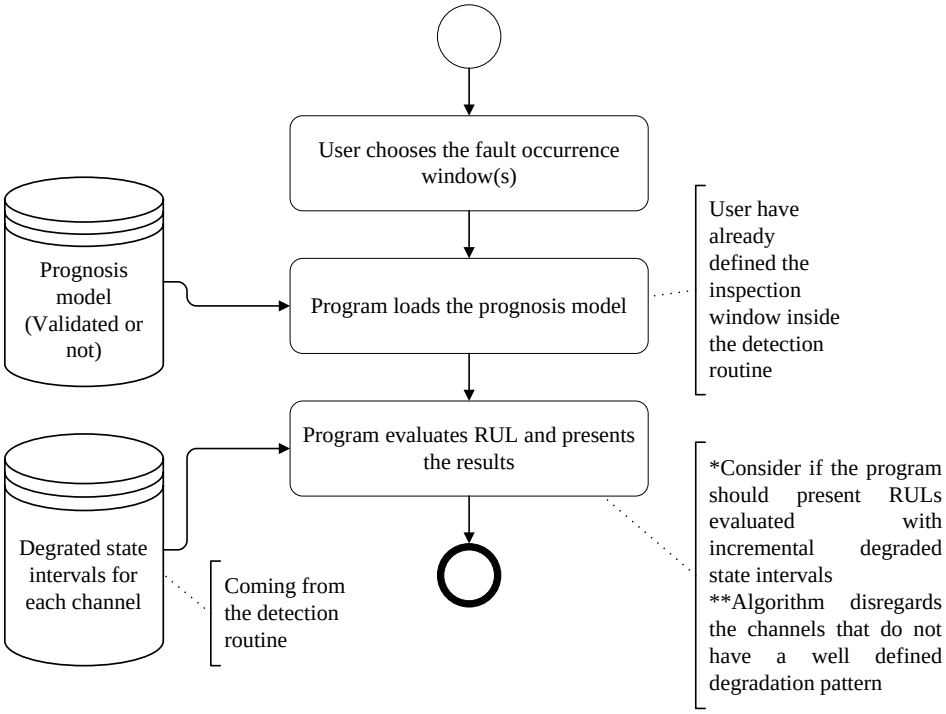
Source: Author

Figure 45: Verification and validation of the prognosis process.



Source: Author

Figure 46: Prognosis routine.



Source: Author