Software Impacts 13 (2022) 100321

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

RUL-RVE: Interpretable assessment of Remaining Useful Life

Nahuel Costa *, Luciano Sánchez

Computer Science Department, University of Oviedo, Gijón, 33202, Asturias, Spain

ARTICLE INFO

Keywords:
Remaining useful life
Prognostics and health management
Interpretability
Variational inference
Recurrent neural networks
Python

ABSTRACT

This paper presents RUL-RVE, a Python tool for the assessment of Remaining Useful Life (RUL). Physical systems are normally subject to degradations that ultimately lead to failure, therefore prognostic technologies are crucial to estimate the lifetime of the system to be monitored. The problem with most existing data-driven approaches is that they lack an explanatory component to understand model learning and/or the nature of the data. RUL-RVE is a framework based on recurrent neural networks and variational inference that can achieve remarkable forecast accuracy while providing an interpretable assessment, which is highly valuable in real-world environments.

Code metadata

Current code version	*v0.1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-58
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/4781584/tree/v1
Legal Code License	MIT License
Code versioning system used	*git*
Software code languages, tools, and services used	*python*
Compilation requirements, operating environments & dependencies	tensorflow>= 2.3.0, matplotlib>= 3.3.4, pandas>= 1.1.5, scikit-learn>= 1.0.2
If available Link to developer documentation/manual	
Support email for questions	costanahuel@uniovi.com

1. Introduction

Remaining useful life (RUL) is an estimate of the length of time an item, component, or system is estimated to be able to function according to its intended purpose before requiring repair or replacement. It is considered a key metric in prognosis that helps improve maintenance schedules and avoid engineering, safety, and reliability failures [1]. This has many real-world applications such as monitoring machining tools, batteries, turbofan engines, and rotating bearings [2].

Many techniques have been proposed to model the degradation of these complex systems, from which two currents arise: model-based approaches and data-driven approaches [3]. The former techniques usually require extensive prior knowledge about the physical systems, information that is often not available in practice. On the contrary, data-driven approaches have become popular in recent years, as they are able to model degradation features based purely on historical records from which the underlying causalities and correlations can be modeled.

The greater impact has undoubtedly been produced by the use of Deep Learning models [4] given that the high dimensionality of raw data obtained from machine health monitoring can be modeled by methods that are known to perform remarkably well, especially in Computer Vision and Natural Language Processing (NLP).

Nevertheless, there is a clear gap between most Deep Learning approaches: although they do achieve very accurate results, models are usually treated as black boxes where it is not trivial to obtain explanations of the decisions that led the model to predict such outputs [5]. Although the current practice is to dispense with prior knowledge about the system to be monitored, in the end, these models are designed to be used by people outside academia. Consequently, it is essential to provide tools that offer some interpretability of the models' decisions,

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: costanahuel@uniovi.es (N. Costa).

https://doi.org/10.1016/j.simpa.2022.100321
Received 27 April 2022; Received in revised form 16 May 2022; Accepted 19 May 2022

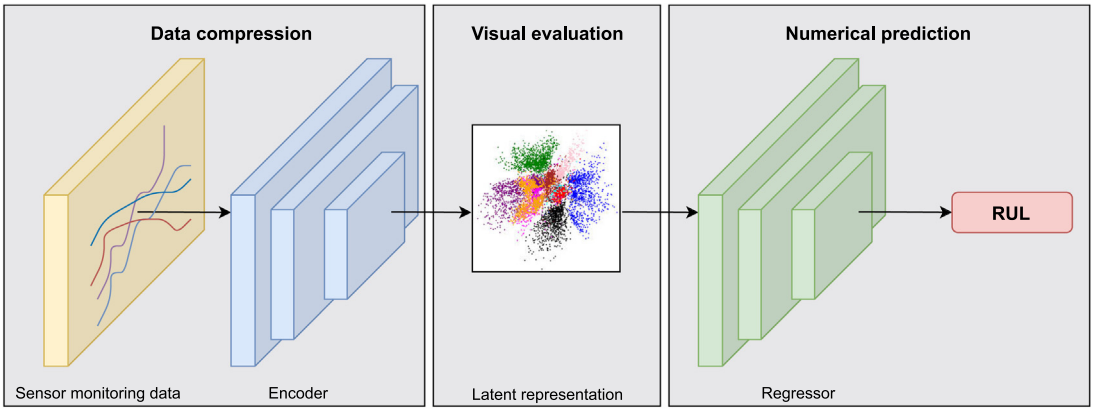

Fig. 1. Workflow followed in the framework: monitoring data is fed into the encoder, which learns a latent representation based on deterioration patterns in order to build a graphical map reflecting the evolution of the samples. The regressor learns from such latent space to report numerically the RUL of each sample.

as well as some insights into the nature of the data. In fact, these are attributes of particular interest, if not demanded, for decision making in safety-critical applications [6].

RUL-RVE [7] is a framework for the task of accurately estimating RUL while providing an explainable and interpretable diagnosis. Its performance was validated on the popular C-MAPSS dataset from NASA [8]. This dataset contains simulated data of Turbofan engines produced by Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), a model-based simulation program. It is composed of multivariate temporal data obtained from twenty-one sensors and is further divided into 4 sub-datasets that differ in operating and fault conditions. RUL-RVE demonstrated that, besides providing a visual assessment of the rate of degradation in aircraft engines, it can also accurately estimate the RUL, where it outperformed most state-of-the-art approaches in terms of RMSE. In addition, its application to data belonging to actual Turbofan engines was also tested to illustrate its performance in a real-world scenario. The model's ability to continuously monitor the useful life of each system makes it possible to detect potential future anomalies, which translated into significant economic savings for the company studied.

2. Description

2.1. Workflow

Fig. 1 illustrates the workflow followed to estimate numerically and visually the RUL. The framework is based on variational inference and is composed of two neural networks: a recurrent encoder and a regression model. The encoder learns to compress the sensor monitoring data to a latent space led by the mean and the variance of an approximated Gaussian distribution, thus resulting in a 2-D representation in which input samples are organized based on their deterioration patterns. The regressor directly influences the training process for obtaining such representation and can also report explicitly which RUL value is the one that best represents each sample that is fed to the model.

In short, the user will have two prediction elements: a numerical estimation of the RUL and a visual map. This allows to visually assess the evolution of the input data, as the samples will be placed in areas close to other samples with similar degradation patterns whose diagnosis is known, thus making it easy to identify those that degrade more rapidly. The fact of having a diagnostic system of these characteristics is crucial for reducing unexpected events to happen because the degradation speed of the components is modeled so that the acceleration in the normal degradation speed of a component can be easily detected.

2.2. Implementation

The RUL-RVE model was implemented in Tensorflow 2 [9]. To introduce variational inference into the loss function, a custom model was created in which the training and testing steps were overridden. Also, a regression penalty was included to cause systems with similar degradation patterns to project into nearby areas of the latent space. Bidirectional LSTM layers were chosen for the encoder as they provide not only information about the past but also about the future, allowing the network to be aware of what the data may look like in its future stages, which helps it to understand what kind of information to predict (different stages of deterioration). For the regressor, a simple Feed Forward network with an intermediate layer with a tanh activation was implemented.

Additional pre-processing methods are included to deal with the most common input data, i.e., multivariate time-series, including scaling, smoothing, window framing and data splitting.

2.3. Usage example

In this section, we provide an example of use of the RUL-RVE to train and test on a given dataset. First, we recommend setting up a new python environment with packages matching the requirements.txt file included in the attached Github repository. It can be easily done with anaconda [10]: conda create –name –file requirements.txt. Another alternative is to run exactly the same environment under which this project was made with Docker [11]. A Docker file is provided, which contains the set of instructions for creating a container with all the necessary packages and dependencies. The fastest way to set it up is to download the project from GitHub, open Visual Studio Code editor, and from the command palette select "Remote-containers: Open folder in Container". Once the environment is configured, within a few lines of code (Fig. 2) the framework can be easily used for training and evaluation.

2.4. Impact overview

Artificial Intelligence research has taken a path in recent years in which the main focus is established on neural networks applied mostly to Natural Language Processing (NLP) and Computer Vision, while the application of some of these powerful algorithms for time series is not yet fully exploited. In addition, the scalability of these models makes the computational requirements increasingly higher, which is a major barrier for most research and industry groups. This is exacerbated by the fact that despite being incredibly good at some tasks, most AI models behave as black boxes that are based on feeding an input to an algorithm that outputs some number or class. However, there are

```python
import model
import utils

# ———————————————— DATA ————————————————
x_train, y_train, x_val, y_val, x_test, y_test = # load your dataset
# ———————————————————————————————————————

# ———————————————— MODEL ————————————————
RVE = model.create_model(timesteps = x_train.shape[1], input_dim = x_train.shape[2],
                         intermediate_dim = 300, batch_size= 128, latent_dim = 2,
                         epochs = 10000, optimizer = 'adam')
# Callbacks for training
model_callbacks = utils.get_callbacks(RVAE, x_train, y_train)
# ———————————————————————————————————————

# ———————————————— TRAINING ————————————————
results = RVE.fit(x_train, y_train, shuffle=True, epochs=epochs,
                  batch_size=batch_size, validation_data= (x_val, y_val),
                  callbacks=model_callbacks, verbose=2)
# ———————————————————————————————————————

# ———————————————— EVALUATION ————————————————
RVAE.load_weights('./checkpoints/checkpoint')
# Visual map
test_mu = utils.viz_latent_space(RVE.encoder, x_test, y_test)
# Evaluate RUL estimation
utils.evaluate(y_test, RVE.regressor.predict(test_mu), 'test')
# ———————————————————————————————————————
```

Fig. 2. Based on a given dataset, the framework is trained and evaluated.

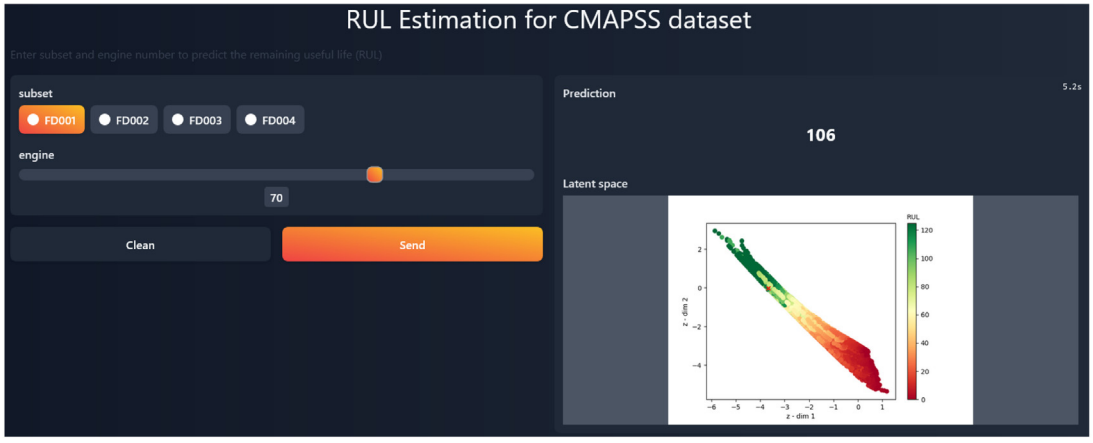

Fig. 3. Gradio demo of the model presented in [7].

many fields where this is not enough and this is where the importance of explainability comes into play. It is crucial to know in some way how these models work internally in order to bring transparency to research and provide certain interpretability of the models' decisions so that they can be easily used by people of any area outside AI.

RUL-RVE emerges as a solution to these problems: it proposes the use of a powerful yet lightweight Deep Learning model: an encoder, implemented with recurrent networks to deal with the temporality of the data, that provides an interpretable evaluation of the component to be monitored. The framework is still young and has been recently published [7] so it has not yet been used in any other existing publications, however, it can be leveraged both in industry and research.

The tool has already been tested in a real industrial test case for RUL prediction of Turbofan engines. This opens up new opportunities to apply the model to other industrial problems of similar nature. Precisely, RUL prediction is present in a wide variety of domains such as manufacturing, power generation, automotive, or transportation. RUL-RVE focuses on the interpretability part to facilitate its use by practitioners in these sectors. The model allows updating the learned patterns as more data becomes available, which makes the successive projections of each system on the visual map form an easily guessable trajectory into the future, something much more valuable than a simple numerical prediction. Also, the lightness of the model in terms of memory facilitates its implementation on any hardware with limited computational capabilities, which is of interest for online monitoring.

On the other hand, since the main value of the tool is to monitor a system in order to detect possible future failures early enough to make decisions to expand its lifetime, RUL-RVE can extend its application to

other contexts. In this sense, the tool is made available to academic researchers so that they can develop and adapt the model to different domains such as health or economics, since the system to be monitored is not limited only to aircraft engines but can also be applied to the stock market or a pacemaker, to name a few examples.

2.5. Illustrative examples

A demo of the model presented in [7] is available at https://huggingface.co/spaces/NahuelCosta/RUL-Variational. The user is presented with the 4 subsets of the CMAPSS dataset, from which they can choose which engine of the test set they want to know the RUL and its representation in the latent space. An illustrative example is shown in Fig. 3. The engine #70 of the FD001 subset is chosen to predict its RUL and visualize its location in the latent space. The RUL estimate is 106 and the location of the engine (marked with an x) is at the top of the map, along with other engines with similar RUL values. The area appears safe, indicating that the engine shows no signs of concern for maintenance at this time.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by the Ministry of Economy, Industry and Competitiveness (“Ministerio de Economía, Industria *y* Competitividad”) from Spain/FEDER under grant PID2020-112726-RB-I00 and by Principado de Asturias, grant SV-PA-21-AYUD/2021/50994.

References

[1] Xiao-Sheng Si, Wenbin Wang, Chang-Hua Hu, Dong-Hua Zhou, Remaining useful life estimation–a review on the statistical data driven approaches, European J. Oper. Res. 213 (1) (2011) 1–14.

[2] Liangwei Zhang, Jing Lin, Bin Liu, Zhicong Zhang, Xiaohui Yan, Muheng Wei, A review on deep learning applications in prognostics and health management, Ieee Access 7 (2019) 162415–162438.

[3] Giduthuri Sateesh Babu, Peilin Zhao, Xiao-Li Li, Deep convolutional neural network based regression approach for estimation of remaining useful life, in: International Conference on Database Systems for Advanced Applications, Springer, 2016, pp. 214–228.

[4] Zhaoyi Xu, Joseph Homer Saleh, Machine learning for reliability engineering and safety applications: Review of current status and future opportunities, Reliab. Eng. Syst. Saf. (2021) 107530.

[5] Davide Castelvecchi, Can we open the black box of AI? Nat. News 538 (7623) (2016) 20.

[6] Enrico Zio, Prognostics and health management (PHM): Where are we and where do we (need to) go in theory and practice, Reliab. Eng. Syst. Saf. 218 (2022) 108119.

[7] Nahuel Costa, Luciano Sánchez, Variational encoding approach for interpretable assessment of remaining useful life estimation, Reliab. Eng. Syst. Saf. 222 (2022) 108353.

[8] A. Saxena, K. Goebel, Phm08 Challenge Data Set, NASA Ames Prognostics Data Repository, NASA Ames Research Center, 2008.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.

[10] Anaconda software distribution, in: Anaconda Documentation, Anaconda Inc., 2020.

[11] Dirk Merkel, Docker: lightweight linux containers for consistent development and deployment, Linux J. 2014 (239) (2014) 2.