Benchmark: Remaining Useful Life Predictor for Aircraft Equipment

Dmitrii Kirov[⋆] and Simone Fulvio Rollini

Collins Aerospace

Abstract. We propose a predictive maintenance application as a benchmark problem for verification of neural networks (VNN). It is a deep learning based estimator of remaining useful life (RUL) of aircraft mechanical components, such as bearings. We implement the estimator as a convolutional neural network. We then provide mathematical formalizations of its non-functional requirements, such as stability and monotonicity, as properties. These properties can be used to assess the applicability and the scalability of existing VNN tools.

URL. Benchmark materials, such as trained models (.onnx), examples of properties (.vnnlib), test datasets, and property generation procedures, are available at `https://github.com/loonwerks/vnncomp2022`.

1 Introduction

Remaining Useful Life (RUL) is a widely used metric in Prognostics and Health Management (PHM) that manifests the remaining lifetime of a component (e.g., mechanical bearing, hydraulic pump, aircraft engine). RUL is used for *condition-based maintenance* to support aircraft maintenance and flight preparation. It contributes to such tasks as augmented manual inspection of components and scheduling of maintenance cycles for components, such as repair or replacement, thus moving from preventive maintenance to predictive maintenance (do maintenance only when needed, based on component's current condition and estimated future condition). This could allow to eliminate or to extend service operations and inspection periods, optimize component servicing (e.g., lubricant replacement), generate inspection and maintenance schedules, and obtain significant cost savings. RUL could also highlight areas for inspection during the next planned maintenance, i.e., it could be used to move up a maintenance/inspection action to prevent component failure. Finally, RUL function can also be used in airborne (in-flight) applications to dynamically inform pilots on the health state of aircraft components during flight.

Multivariate time series data is often used as RUL function input, for example, measurements from a set of sensors monitoring the component state, taken at several subsequent time steps (within a time window). Additional inputs may include information about current flight phase, mission and environment. Such

⋆ Corresponding author. Email: dmitrii.kirov@collins.com highly multi-dimensional input space motivates the use of Machine Learning (ML), more precisely, Deep Learning (DL) solutions with their capabilities of performing automatic feature extraction from raw data. A number of solutions based on Deep Neural Networks (DNNs) has emerged over recent years (e.g., [4], [5], [6]).

An example of a DL-based RUL function is illustrated in Figure 1. The DL model is a Convolutional Neural Network (CNN) that accepts as input a time window of length 40, i.e., snapshots reflecting the bearing state taken at 40 consecutive time steps. Each time step corresponds to 1 hour. Each snapshot contains the following information:

- Seven Condition Indicators (CIs) that provide numerical information about the bearing degradation, obtained from signal processing of measurements of the vibration sensor attached to the bearing.
- Information about the current mission: current flight regime, such as ascent, cruise and descent, with corresponding nominal component load (bearing RUL heavily depends on how the component is loaded), mission type (a set of predefined mission patterns is available, e.g., long, short, mixed).
- Information about the current flight environment (e.g., desert or non-desert).

CI values are computed by the Health Usage Monitoring System (HUMS), while remaining inputs are provided by other avionics software. Snapshots are provided to the input generator, where they are stored in memory and periodically shifted to yield a new consecutive time window. Output of the CNN is a numerical non-negative value that represents the bearing RUL in hours. It is provided to the end users, such as pilots and MRO (Maintenance, Repair and Overhaul) via cockpit and ground station displays. Additional pre- and post-processing of the CNN (e.g., input normalization) is not described due to space limitations, details are available in [2].

2 Model Description

We propose a Convolutional Neural Network (CNN) model adapted from [1] as a benchmark for the competition. It corresponds to the use case described above. The CNN accepts as input a sequence (time window) of inputs. The inputs are snapshots of condition indicators at a given time step, as well as other metrics. Several convolutional layers are used to apply 1D convolutions to the inputs along the time sequence direction. Extracted features are then merged together via a fully connected layer. Dropout is used to mitigate overfitting. Activation functions at all layers are Rectified Linear Units (ReLUs). The CNN performs a regression task and outputs a numerical value, which is the RUL.

Several neural networks of different complexity are provided:

- **NN_rul_small_window20.onnx** (number of ReLUs - 5500)
- **NN_rul_full_window20.onnx** (number of ReLUs - 10300)
- **NN_rul_full_window40.onnx** (number of ReLUs - 28300)

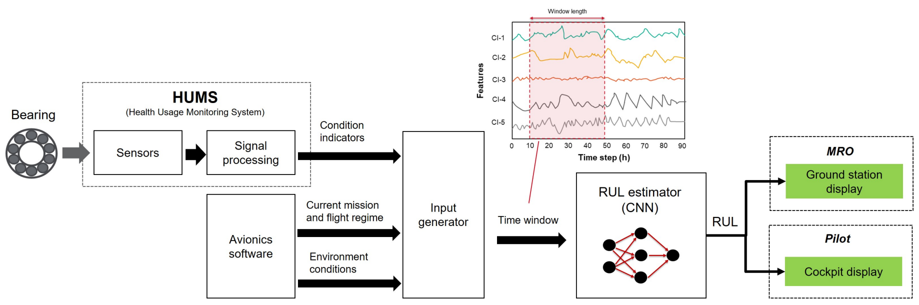

Fig. 1. Overview of the RUL estimator and its context.

All networks have been trained using the same dataset. The motivation for providing several networks is the scalability study. The number of ReLUs is different for each network. This seems to be one of the key complexity metrics for many VNN tools. Internally, networks have the same architecture/layers, but some different hyperparameters, such as the number of filters in convolutional layers ("small" networks have fewer filters, "full" networks have more). Also, two window sizes (20 and 40) have been used to generate the networks. While the number of input features remains constant for all networks, manipulating the window size allows to change the input space size (2x in the case of window sizes used). Change in window size has a more significant impact on the overall CNN complexity.

3 Properties Description

We propose several classes of properties for the NN-based RUL estimation function. First two classes (stability and monotonicity) are *local*, i.e., defined around a given point. To address the requirement for randomizing the inputs to VNN tools, we provide a script with adjustable random seed that can generate these properties around input points randomly picked from a test dataset. Properties of the last class ("if-then") are defined over input ranges. We have generated a list of such properties and provide means for randomly selecting properties from this list. Below, a short description of each property class is provided.

3.1 Stability Properties

ML model is *stable* if a small, bounded perturbation applied to its inputs in normal operating conditions, i.e., when the inputs are inside the ML model's operational design domain, does not cause a significant deviation in its output. Here, we focus on perturbations of condition indicator (CI) inputs. Corresponding model stability requirement could be formulated as follows: *"For a simultaneous perturbation of one or many CIs at a single time step (e.g., due to a resonance frequency) within any time window, the output deviation of the RUL* *estimator shall not exceed ϵ hours. The maximum admissible perturbation that can occur to a CI input shall be equal to δ% of the average initial value of that CI that corresponds to a fully healthy state of the bearing component".* Then, local stability properties that consider perturbations of one or more inputs can be expressed in the "delta-epsilon" form:

$$\forall \mathbf{x}' : \forall i \in S : |x_i'^t - x_i^t| < \delta_i |x_i^*| \implies |f(x') - f(x)| < \epsilon, \tag{1}$$

where prime ($'$) denotes a *perturbed* item (i.e., $\mathbf{x}'$ is the time window where one or more elements have been perturbed; $x_i'^t$ is a perturbed value of the CI i at time step t; $f(\mathbf{x}')$ is the ML model output computed from the perturbed input), x_i^* represents the average initial value of the CI i, computed over all data, S is a subset of the indexes corresponding to perturbed CIs, δ_i is the bound on the input perturbation for a chosen CI i *w.r.t.* x_i^* (expressed as a percentage), and ϵ is the maximum admissible output change.

The property requires that for any input perturbation applied to CIs from the set S, bounded by a corresponding δ_i, the output must not deviate by more than ϵ (L_∞ norm is used to define perturbations). Stability properties provided in this benchmark differ in the number of perturbed CIs and perturbation magnitudes.

3.2 Monotonicity Properties

Differently from stability requirements, the requirements on monotonicity of the RUL estimator concern all steps of the input window, i.e., within the entire window. This is a realistic situation that may occur, for example, due to damage or excessive load in the bearing that leads to an increased degradation rate. Therefore, such changes in the CIs over the entire time window shall not be identified by data quality indicators in the HUMS as abnormal (unlike random spikes/perturbations occurring at multiple time steps). Monotonicity requirements prescribe a non-increasing behavior of the estimator given a change in the growth rate (higher or smaller) of one or more condition indicators.

Condition indicators are correlated with component degradation and failures. Their values are expected to *monotonically increase* during the use of the bearing component, which reflects its degradation. Consequently, the bearing RUL is expected to *monotonically decrease.* Expected behavior of the RUL estimator output is monotonic with respect to the inputs (CIs), i.e., when CIs increase the RUL should decrease.

A growth rate increase of a CI by some percentage represents a different CI "trajectory" within the time window. Such new trajectory can be used as an upper bound, while the original CI growth trend represents a lower bound. The verification strategy is to analyze all possible CI trajectories within these bounds. Let $\mathbf{x_i}$ be the vector of values of some CI i in the time window, x_i^t being the CI value at time step t. Local monotonicity property for this CI can be formulated as

$$\forall \mathbf{x}' : \forall t : x_i^t \leq x_i'^t \leq x_i^t + \gamma |x_i^t - x_i^1| \implies f(\mathbf{x}') \leq f(\mathbf{x}) \tag{2}$$

where prime ($'$) denotes a modified item (i.e., $\mathbf{x}'$ is the time window, where one or more CIs have modified growth rates *w.r.t.* the original time window $\mathbf{x}$; $x_i'^t$ is a modified value of the CI i at time step t; $f(\mathbf{x}')$ is the ML model output computed from the modified input), and γ (gamma) is the parameter that regulates the CI slope change.

This property states that for any growing CI trajectory bounded by the original CI trajectory (lower bound) and the one changed by a percentage γ (upper bound), i.e., a steeper growth trend, the RUL shall be non-increasing *w.r.t.* the original CI trajectory. The difference $|x_i^t - x_i^1|$ represents an approximation of the CI slope in the interval $[1, t]$. Further discussion on formalization of monotonicity properties for the RUL is available in [2].

3.3 If-Then Properties

These properties are formulated as follows: IF the CNN inputs are in given ranges, THEN the output (RUL) must be in an expected range. To generate such properties, input ranges of some inputs have been broken down into several sub-ranges (e.g., Low, Medium, High) with corresponding lower and upper bounds. For each combination of these sub-ranges, an expected output range has been estimated. Given these numerical bounds on input/output ranges, if-then properties are straightforward to formulate.

NOTE. Given the number of combinations of input ranges, the number of if-then properties can be extremely large. Therefore, for this benchmark we have generated only a small subset of such properties with different complexity, based on size and the number of the ranges. The properties are randomly selected from this subset. If-Then properties are currently the hardest to verify.

4 Concluding Remarks

We have proposed a benchmark problem for neural network verification tools – a convolutional neural network that predicts remaining useful life of a mechanical component. Property verification an important capability in learning assurance. European Union Aviation Safety Agency (EASA) in their AI Concept Paper [3] emphasizes Formal Methods as anticipated means of compliance for the verification of ML models and their properties, such as robustness.

Acknowledgements The authors wish to thank Eric DeWind and David F. Larsen for fruitful discussions and feedback about the RUL estimator and its properties, as well as for providing mechanical bearing degradation datasets to prepare this benchmark problem.

References

1. Remaining Useful Life Estimation using Convolutional Neural Network. [Online]. https://www.mathworks.com/help/releases/R2021a/predmaint/ug/remaining-useful-life-estimation-using-convolutional-neural-network.html

2. EASA and Collins Aerospace: Formal Methods use for Learning Assurance (ForMuLA). Tech. rep. (April 2023)
3. European Union Aviation Safety Agency (EASA): Concept Paper: Guidance for Level 1&2 Machine Learning Applications. Concept paper for consultation. (February 2023)
4. Li, X., Ding, Q., Sun, J.Q.: Remaining useful life estimation in prognostics using deep convolution neural networks. Reliability Engineering & System Safety pp. 1–11 (2018)
5. Ren, L., Cui, J., Sun, Y., Cheng, X.: Multi-bearing remaining useful life collaborative prediction: A deep learning approach. Journal of Manufacturing Systems **43**, 248–256 (2017)
6. Yuan, M., Wu, Y., Lin, L.: Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In: 2016 IEEE international conference on aircraft utility systems (AUS). pp. 135–140. IEEE (2016)