Operating Condition-Invariant Neural Network-based Prognostics Methods applied on Turbofan Aircraft Engines

Gabriel Pasa[1], Ivo Medeiros[2], and Takashi Yoneyama[3]

[1,2] *Embraer, São José dos Campos, São Paulo, 12227-901, Brazil*
gabriel.pasa@embraer.com.br
ivo.medeiros@embraer.com.br

[3] *Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, 12228-900, Brazil*
takashi@ita.br

ABSTRACT

Neural networks in their many flavors have been widely used in prognostics of engineered systems due to their versatility and increasing potential, especially with recent breakthroughs in Deep Learning and specialized architectures. Despite these advances, some problems can still significantly benefit from a solid exploratory analysis and simple task-specific data/target transformations. In this work, popular architectures including Feedforward, Convolutional and LSTM (Long Short-Term Memory) networks are evaluated in a case study of RUL (Remaining Useful Life) prediction for turbofan aircraft engines, using data from publicly available repositories. A robust set of over $20,000$ model configurations are tested, evaluating the effects of several hyper-parameters and design choices. The latter includes a maximum prediction horizon, revealing a trade-off between prediction accuracy and timeliness which can have significant impact in real-world applications. An operating condition-specific standardization scheme is also evaluated, in order to minimize the impact of normal changes in operating regimes which obfuscate the fault degradation patterns. A comparison with existing works in literature shows some simple policies for operating condition-invariance have lead to results which outperform the current state-of-the-art methods for some of the data subsets with multiple operating conditions.

1. INTRODUCTION

Engineers constantly aim to design systems increasingly performant, but also capable of remaining so for long periods of time. Specifically, in the aerospace industry, reliability and availability are very important traits due to the high associated acquisition, repair and operation costs for aircraft. The aerospace industry (like the automotive, data center, oil and gas, steel-making, etc.) has high stakes and potentially catastrophic consequences for failure, so a substantial effort is put into reliability and maintainability, from concept design to product end-of-life.

Prognostics and Health Management (PHM) technologies are a core enabler for predictive maintenance, allowing for timely interventions that prevent unscheduled downtime, helping in the decision-making process, making operation not only safer but also more cost effective.

The focus of this work is on prognostics, or remaining useful life (RUL) prediction, more specifically machine learning data-driven methods based on neural networks. Publicly available data sets describing the operational history of simulated aircraft turbofan engines are used in the case study, allowing for comparisons with other results from the literature. The explored architectures include multilayer perceptron (MLP), convolutional neural networks (CNN) and long short-time memory (LSTM) networks. The analysis covers the required steps for implementing such data-driven prognostics techniques, including exploratory analysis, data preprocessing, feature engineering and selection, model structure and hyperparameter optimization.

2. LITERATURE REVIEW

2.1. Taxonomy of Prognostics Approaches

Although no universal consensus seems to exist on how to best categorize the different prognostics methods, a useful classification is shown in (Javed, Gouriveau, & Zerhouni, 2017):

Physics-based methods: also known as model-based methods, they use explicit mathematical formulations to represent the system behavior (Pecht & Jaai, 2010). These tend to be more accurate, but require in-depth knowledge about the physical processes that govern the system behavior and assume it can be described analytically and accurately, making them not always possible or cost effective.

Data-driven methods: these methods use black box models, learning the system behavior via condition monitoring data, not requiring extensive expect knowledge about the system. The use of general-purpose models also allows for more flexibility, allowing applicability across different contexts, but accuracy is dependent on the quantity and quality of available data (Ahmadzadeh & Lundberg, 2013).

Hybrid methods: these approaches leverage aspects of both physics-based and data-driven methods, using them in complementary ways.

In this work, all studied methods fall into the second category, the data-driven approaches. Those represent a good compromise between accuracy and applicability, especially in times of ubiquitous and abundant data sources and the technological advances in machine learning / deep learning, which show tremendous potential in many fields.

2.2. Machine Learning-based Prognostics Methods

There are a number of papers in the literature proposing data-driven methods for health management-related tasks using machine learning, statistical, ad-hoc methods, etc. Upon review, some similarities among the different prognostics approaches can be noticed.The concept of a health index (HI) is used by many authors. The problem is then broken into two steps, one to map the predictor features into the HI and the other to map the HI to the RUL. Examples of works using HI's include (Yang et al., 2016), (Wang, Yu, Siegel, & Lee, 2008) and (Peng, Xu, Liu, & Peng, 2012).

When dealing with measurements of physical quantities, sensor noise will always be present to some extent. This is addressed in multiple studies by applying some form of pre/post-processing smoothing filter. In (Yang et al., 2016), a dynamic filter is applied to the HI, enforcing monotonicity and gradualness (health will generally only slightly decrease over time). Similarly, a moving average filter is used in (Wang et al., 2008) and a kernel filter in (Ramasso, 2014). Several different neural network architectures have been used for prognostics, such as feedforward networks in (Yang et al., 2016), CNN's in (Li, Ding, & Sun, 2018), recurrent neural networks (RNN) are used in (Liu, Saxena, Goebel, Saha, & Wang, 2010), LSTM networks are used in (Guo, Li, Jia, Lei, & Lin, 2017), to name a few.

3. METHODOLOGY

3.1. Problem Overview

The CMAPSS (Commercial Modular Aero-Propulsion System Simulation) data sets are used in this work. These are divided into TEDS (Turbofan Engine Degradation Simulation) (Saxena & Goebel, 2008b) and PHM2008 (Saxena & Goebel, 2008a) data sets, which are both publicly available at the NASA Prognostics Data Repository.

The data is comprised of several multivariate time series, which correspond to 24 sensor measurements taken for each operating cycle of a particular simulated turbofan engine. Over time, the engines start to degrade and the goal is for the model to predict the number of cycles until failure (RUL). As seen in Table 1, each data set is divided into subsets, which may have different numbers of fault modes and operating conditions.

Table 1. Overview of the CMAPSS datasets subsets.

Dataset	Subsets	# Fault Modes	# Operating Conditions	# Train Units	# Test Units
TEDS	FD001	1	1	100	100
	FD002	1	6	260	259
	FD003	2	1	100	100
	FD004	2	6	249	248
PHM2008	FD005	1	6	218	218
	FD005 Final	1	6		435

In the training set, all time series are assumed to go on until failure. In the test set, the data stops before that happens. In the TEDS data set, ground truth for the number of cycles left is provided. In the PHM2008, it is not, so one must submit their result to the repository website to get the results.

Figure 1 shows the windowing scheme used to map segments with length ℓ from the input time series $\boldsymbol{X}^{(u)}$ from the u-th engine unit into a scalar $\boldsymbol{y}^{(u)}$ representing the number of flight cycles left, which decreases linearly over time. This relationship is what needs to be learned by the models.

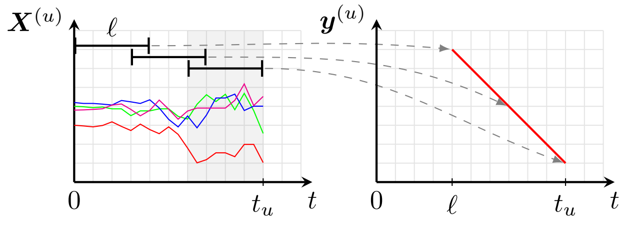
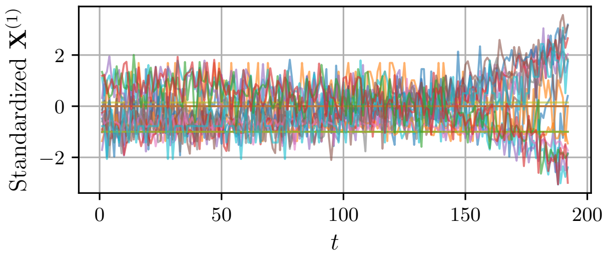


Figure 1. Moving window sequences to RUL curve mapping.

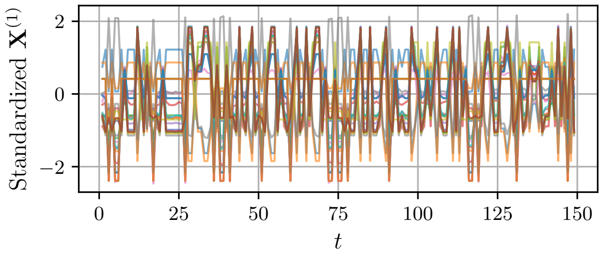
3.2. Exploratory Analysis

By plotting the normalized sensor data, as seen in Fig. 2a, it can be seen that different features behave differently, but a clear fault degradation trend can be seen for some of them. However, in the case where there are multiple operating conditions, as in Fig. 2b, the same degradation trend is not clearly visible. It turns out that the variations in the predictor features caused by changes in the operating conditions are way more pronounced than the variations due to the degradation.

Since that degradation pattern is what needs to be recognized in order to predict RUL, the second case is considerably more complex to model.



(a) Single operating condition.



(b) Multiple operating conditions.

Figure 2. Plots of the time series data from two arbitrary units ($u = 1$) from subsets FD001 and FD002, which have, respectively, one and six operating conditions.

It is stated at the problem description that the first three features encode the operating conditions. When plotting those features only, 6 well defined clusters can be seen, as shown in Fig. 3 (TEDS and PHM2008 data sets have different operating conditions).

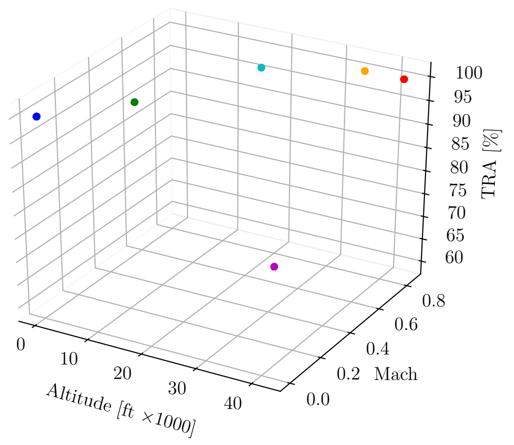
3.3. Data Preprocessing

Several data preprocessing strategies are evaluated in this work. Feature selection is performed manually, by visually inspecting the behaviors of the individual features over time.

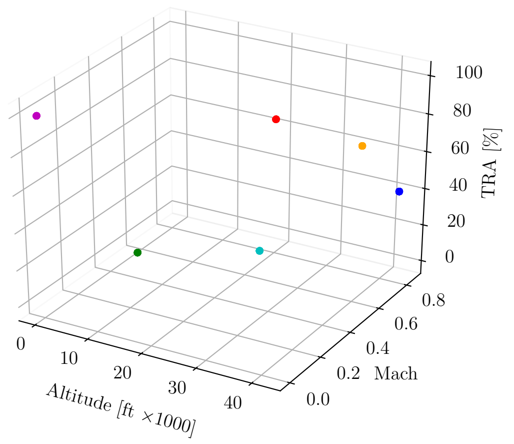
Three scenarios are evaluated: one where all features are kept, one where the features not directly correlated with RUL are removed and one where most features are removed, leaving only a few which do not seem to contain redundant information (see Table 2 for details).

3.3.1. Operating Condition-specific Standardization

Feature scaling is generally desirable in order to improve convergence of gradient-based learning algorithms, such as the ones used to train neural networks. Feature transformation was initially done by performing regular feature-wise standardization, as seen in Eq. (1), where $\tilde{\boldsymbol{X}}^{(u)}_{:,d}$ is the standardized d-th feature from the u-th engine unit, $\boldsymbol{X}^{(u)}_{:,d}$ is the original data, μ_d is the mean from the d-th feature and σ_d is the standard deviation.



(a) TEDS.



(b) PHM2008.

Figure 3. Clusters found in the features describing the engine operating conditions in CMAPSS dataset (including all subsets).

$$\tilde{\boldsymbol{X}}^{(u)}_{:,d} = \frac{\boldsymbol{X}^{(u)}_{:,d} - \mu_d}{\sigma_d} \tag{1}$$

However, it was noticed that the operating conditions have a great impact on the predictors, as seen in Fig. 2. In fact, the (normal and expected) variance observed as a consequence of different operating conditions far outweighs the variance introduced by the progressing faults. This essentially masks the fault's effects making it more difficult to predict the RUL if multiple operating conditions are present. Therefore, an effort was made to make the data operating condition-invariant.

This is achieved by applying the operating condition-specific standardization method described by Eq. (2) and Eq. (3) in all features correlated with RUL. Similar feature standardization schemes have been proposed by other authors, such as (Peel, 2008) and (Wang, 2010), with good results.

$$\tilde{\boldsymbol{X}}^{(u)}_{:,d} = \sum_{c=1}^{n_c} \boldsymbol{\delta}^{(u)}_{:,c} \odot \left(\frac{\boldsymbol{X}^{(u)}_{:,d} - \mu_{d,c}}{\sigma_{d,c}} \right) \tag{2}$$

$$\delta^{(u)}_{t,c} = \begin{cases} 1, & \text{if } c = \arg\min_{c'} \sqrt{\sum_{d=1}^{3} \left(X^{(u)}_{t,d} - \mu_{d,c'} \right)^2} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Since each operating condition corresponds to a cluster in the $\boldsymbol{X}_{:,1:3}$ subspace (subspace defined by the first three features), a simple clustering algorithm (K-Means) is used to assign each data point to a cluster via a hard membership function $\delta^{(u)}_{t,c}$. Each data point is then standardized using the statistics drawn from only the samples coming from the same cluster it belongs to. Similarly, $\mu_{d,c}$ and $\sigma_{d,c}$ are the mean and standard deviation, respectively, of the d-th feature considering only samples belonging to the c-th cluster. The number of clusters n_c corresponds to the number of operating conditions in the data subset.

3.3.2. Noise Filtering

When analyzing the top ranked prognostics algorithms for this problem, it can be seen that all of them use some form of filter in order to decrease the effects of sensor noise. In this work a causal exponentially weighted filter, described by Eq. (4), is used to that end.

$$\tilde{X}^{(u)}_{t,d} = \begin{cases} X^{(u)}_{t,d}, & \text{if } t = 0 \\ \alpha \cdot X^{(u)}_{t,d} + (1-\alpha) \cdot \tilde{X}^{(u)}_{t-1,d}, & \text{otherwise} \end{cases} \tag{4}$$

$X^{(u)}_{t,d}$ is the original observation at time t and $\tilde{X}^{(u)}_{t,d}$ is the resulting filtered value. Low values of the parameter α lead to more pronounced filtering, but also a longer delay is introduced in the signal.

3.3.3. RUL Limit ($r_{\max}$)

Since the target RUL value is only given for the final observation in the time series, targets for earlier windows may be defined arbitrarily, not necessarily assuming a linearly decreasing trend (as seen in Figure 1) throughout the entire engine unit's life.

A simple target transformation is used, in order to limit the maximum RUL ($r_{\max}$) value in the target RUL curve. Three scenarios are evaluated, setting this cap to 130 (most common among the works in the literature), 100 and 80 cycles. Going back in time far enough will likely eventually make one reach a stage at which no fault is present, the system is healthy with no signs of degradation, making it unreasonable to expect the model to predict a future failure then. Introducing this RUL limit reduces the uncertainty of early predictions, making the models more reliable.

3.4. Model Architectures

In this work, three neural network predictive model architectures are evaluated. The first is the classic Multilayer Perceptron (MLP). The only noteworthy adaptation that was made necessary was the flattening of the input multivariate time series into a single univariate array, by concatenating the feature vectors. This was necessary since the MLP structure only supports unidimensional inputs.

The second architecture is the CNN. Since the feature vectors are relatively small, there was no need to include pooling layers in between the convolutional ones. For all the tested models, at least one fully connected layer was used at the end of the network.

The third architecture is the LSTM. No special adaptation was necessary.

For all architectures, several structures were tested, changing the number of layers, width per layer, filter lengths (CNN), etc. The ranges of values experimented with were determined in an attempt to equalize the average training time for each the architectures (see Table 2 for details).

3.5. Training and Evaluation

The Adam (adaptive moment estimation) optimizer (Kingma & Ba, 2014) was used for all experiments. The original training set was broken into 80% for training and 20% for validation. The original test set was reserved for testing. As for loss functions, initially, a simple mean squared error (MSE) was considered, as described in Eq. (5), where n_y is the number of observations, $\hat{y}_i$ and y_i are the i-th predicted and target outputs, respectively (i being a simple index, not necessarily time).

$$\text{MSE} = \frac{1}{n_y} \sum_{i=1}^{n_y} (\hat{y}_i - y_i)^2 \tag{5}$$

Since for the prognostics task a late prediction is more harmful than an early one, an asymmetric score s was devised by the creators of the dataset in order to penalize late predictions more strongly, as described by Eq. (6).

$$s = \begin{cases} \sum_{i=1}^{n_y} \left[\exp\left(-\frac{\hat{y}_i - y_i}{10} \right) - 1 \right], & \text{for } \hat{y}_i < y_i \\ \sum_{i=1}^{n_y} \left[\exp\left(-\frac{\hat{y}_i - y_i}{13} \right) - 1 \right], & \text{for } \hat{y}_i \geq y_i \end{cases} \tag{6}$$

Note, however, that the Eq. (6) does not normalize the score by the number of observations accounted for, which makes it unsuitable for comparisons between data sets with different sample sizes. In order to overcome this inconvenience, the normalized score $\tilde{s}$ described by Eq. (7) was used as a loss function instead.

$$\tilde{s} = \begin{cases} \frac{1}{n_y} \sum_{i=1}^{n_y} \left[\exp\left(-\frac{\hat{y}_i - y_i}{10} \right) - 1 \right], & \text{for } \hat{y}_i < y_i \\ \frac{1}{n_y} \sum_{i=1}^{n_y} \left[\exp\left(-\frac{\hat{y}_i - y_i}{13} \right) - 1 \right], & \text{for } \hat{y}_i \geq y_i \end{cases} \tag{7}$$

Table 2. Summary of the hyperparameter values and design choices explored through random search.

Hyperparameter	**Search space**

Selected features	Kept all;
	Removed 4, 8, 19, 21 and 22;
	Only kept 7, 14, 16, 18 and 24

Op. cond. as one-hot features	Yes or No
RUL limit ($r_{\max}$)	130, 100, 80
Sequence length	20, 21, ..., 40
Batch size	32, 64, 128, 256, 512, 1024
Op. cond.-specific standardization	Yes or No
Exponential filter α	0.001 to 0.2 (log distribution)
Extra fully-connected layer	No extra layer, 64 or 128 neurons
Hidden Layer activation function	Tanh, RELU, Sigmoid
Learning rate	0.0001 to 0.01 (log distribution)
Dropout probability	0 to 0.6
LSTM units per layer	32, 64, 128, 256
LSTM layers	1 or 2
CNN filters	4, 8, 16, 32, 64, 128, 256, 512
CNN kernel size	3, 4, 5, ..., 11
CNN layers	1, 2, 3, 4, 5
MLP units per layer	16, 32, 64, 128, 256, 512, 1024
MLP layers	1, 2, 3, 4
Number of epochs	20 for random search; 50 to reproduce best results
Optimizer	Adam
Loss function	Normalized score ($\tilde{s}$)

3.6. Hyperparameter Search

Several methods for hyperparameter optimization were considered in this work, but random search was eventually chosen. This method is very simple, computationally cheap, explores the search space relatively well despite the presence of unimportant hyperparameters. Other more complex optimization strategies, such as genetic algorithms for example, may yield better results but also introduce other hyperparameters themselves.

Table 2 shows a list of all hyperparameters / design choices and their values explored during random search. In all architectures, the layer width is increased by a factor of 2 as depth is increased (i.e.: if the first layer has width 64, the second will have 128, the third 256, etc.).

4. EXPERIMENTAL RESULTS

4.1. Random Search Results

During random search, over $20,000$ different models were evaluated, being trained with 20 epochs. Figure 4 shows the results of these model evaluations (each point is a model evaluation), in light of the MSE and normalized score for the validation set.

From the results in Figure 4b it can be seen that a lower RUL limit will allow the models to achieve lower MSE's and scores. This poses an interesting trade-off between prediction accuracy and timeliness. If the model is forced to make too early predictions, its uncertainty is higher, so the error tends to increase. However, if the limit is too low, the model may only give non-trivial predictions (predictions below $r_{\max}$) when it is too late to take any action.

The first plot in Figure 5 shows the average output from the best models (in terms of normalized score) from each architecture (MLP, CNN and LSTM) for each engine unit in the validation set of subset FD004. The shaded areas represent the standard deviation for predictions at each time step. The dashed lines show the target RUL, limited by $r_{\max}$. The second plot shows the mean squared error of predictions over time and the third shows the number of data points present in each time step. Data points come from each of the three evaluated models for each of the validation set data sequences for each of the time steps.

In the second plot of Figure 5 it can be seen that the error is generally higher for higher values of $r_{\max}$ for pretty much the entire time span featured in the datasets. A peak in the errors can be seen at the discontinuity between the constant and linearly decreasing regimes. The difference between the errors for different values of $r_{\max}$ also becomes much smaller after this discontinuity (as target RULs coincide), but is still present.

RUL predictions for larger values of $r_{\max}$ in the constant target regime tend are significantly underestimated, which is a consequence of the asymmetrical loss function that penalizes late predictions and the larger uncertainty for earlier degradation states.

In the first plot of Figure 5 it can be seen that the models with higher RUL limits tend to underestimate the RUL value. This is due to the asymmetrical loss function described in Eq. (7).

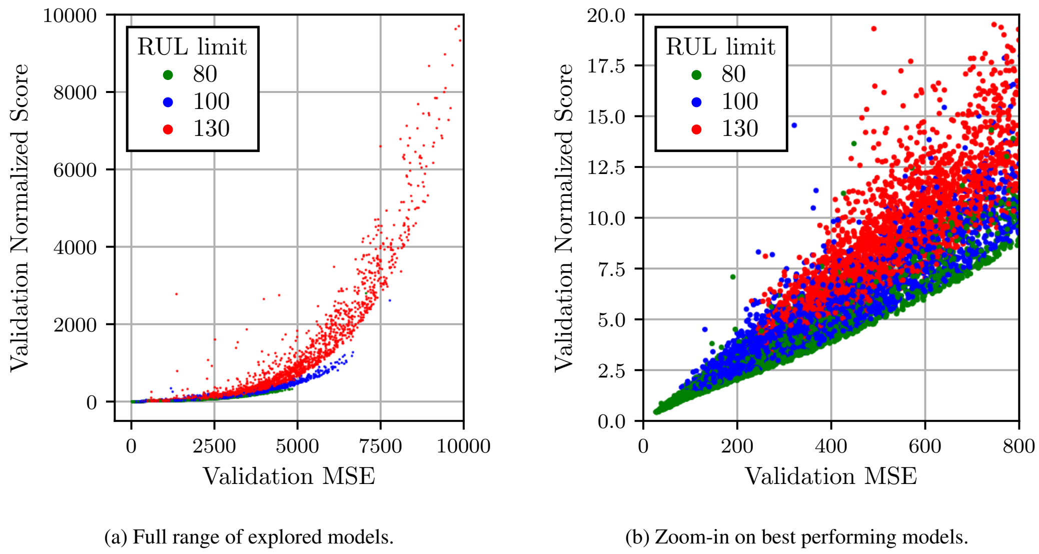


Figure 4. Scatter plot of normalized scores (equation 7) and MSE (equation 5) on the validation set for all CMAPSS subsets (FD001 to FD005) from model configurations obtained by random search. Color coded by RUL limit ($r_{\max}$).

The behavior for the other subsets was very similar to that, which is why it was omitted.

4.1.1. Hyperparameter Effects

By analyzing the many models generated during random search it is possible to assess the overall effect each parameter has on average on the prediction results. Since the hyperparameter values are chosen at random (in this case mostly drawn from uniform distributions), the effects from each parameter are not seen in isolation. Instead, what is seen is the withstanding effect despite changes in other parameters.

Figure 6 shows the impact the operating condition-specific standardization has on the aggregated validation results from all data subsets and all model architectures. On average, the normalized score tends to be lower if this method is used, surely a consequence of the better results from subsets FD002, FD004 and FD005, which have multiple operating conditions.

Similarly, Figure 7 shows the impact of the choice of $r_{\max}$. The lower the limit, the better the scores, on average, which is consistent with previous observations regarding the accuracy and timeliness trade-off.

Figures 8 and 9 show the effects of more structural hyperparameters, specifically the model width (number of neurons per layer) and model depth (number of layers). Neurons and layers in different architectures are not necessarily comparable, for instance the computational cost of an LSTM layer [1] is significantly higher than an MLP layer with the same width. The ranges for depth and width for each architecture were chosen by hand, in a way to make models converge to non-trivial (not overfit or underfit) solutions and to make training times similar (fair division of computational time).

Note that the width shown in Figure 8 is taken on the narrowest layer, and the width increases with layer depth. On average, models seem to benefit from having wider layers, however the training time and chance of overfitting will increase, so the trend seen in 8 will not continue indefinitely as width increases (even wider models were not trained to restrain the computational cost and training times).

It can be seen on Figure 9 that on average, relatively shallow models with 2 or 3 layers are easier to train, and are able to reach good scores. It is easier to have converging models with less layers than with many layers, even if the overall performance of the deeper models that do converge may be better.

4.2. Comparison with Literature Results

As pointed out in (Ramasso & Saxena, 2014), many authors have used the CMAPSS datasets in their work, making it an opportunity to further validate the achieved results in a benchmark. Tables 3 and 4 show a comparison between the results from this work and results from the literature for the subsets in the TEDS dataset, for RMSE (root MSE), Eq. (5), and asymmetric score, Eq. (6), respectively. Ideally more evaluation metrics could be used, such as the ones described in

[1]In this work what is referred to as "LSTM layers" are the entire recurrent LSTM units, which can be stacked in series (output of the first layer connected to the input of the next, and so on), not the internal subcomponents of the LSTM cell.

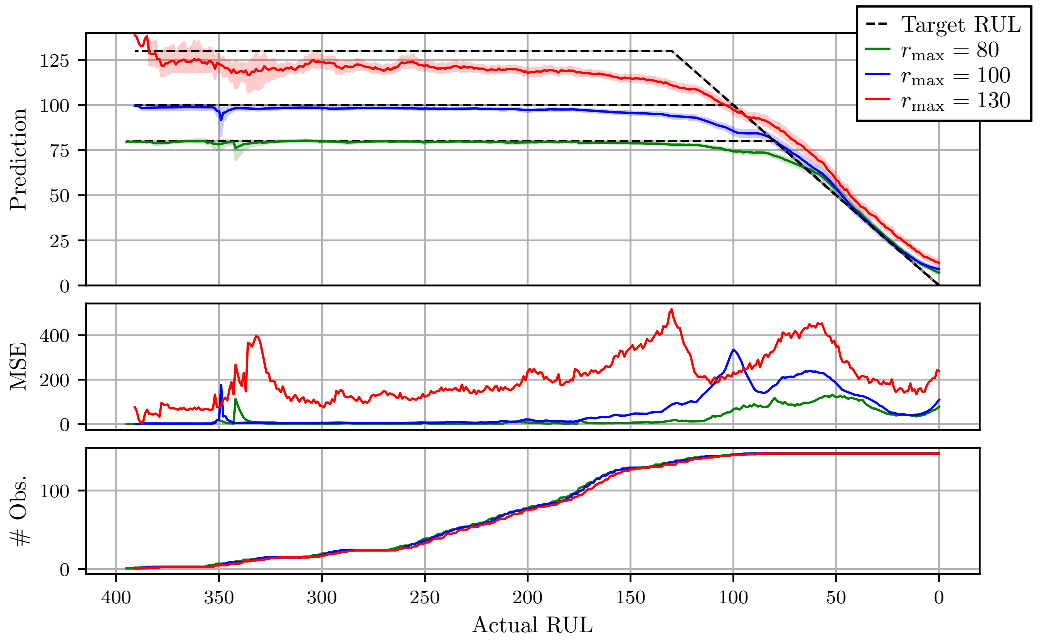


Figure 5. Prediction results for the validation set of subset FD004. Each time series describes the average results for the set of best models (best MLP, CNN and LSTM model in terms of normalized score) for each value of $r_{\max}$, across all engine units in the validation set. First plot shows the actual RUL prediction, with the standard deviation highlighted. Second and third plots show the average MSE and number of available observations over time.

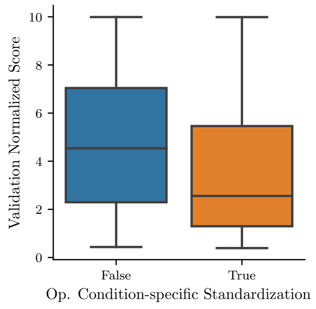


Figure 6. Effect of op. condition-specific standardization.

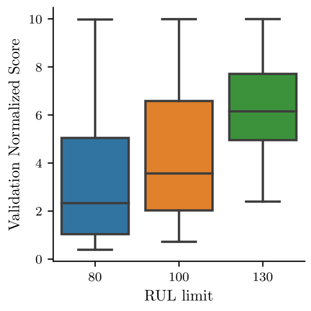


Figure 7. Effect of $r_{\max}$

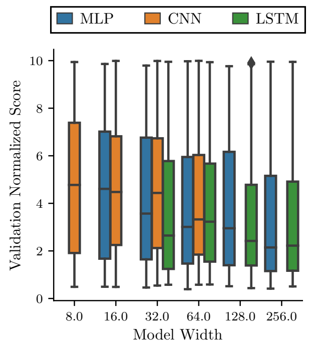


Figure 8. Effect of model width.

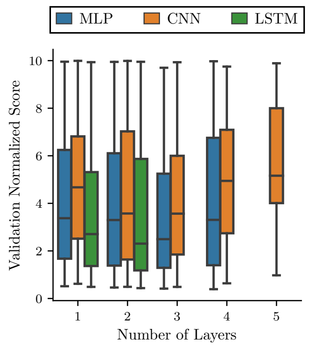


Figure 9. Effect of model depth.

(Saxena et al., 2008), but RMSE and the asymmetric score are more widely used for this problem, being a common metric for comparison between works.

By analyzing Table 3, it can be seen that for subsets FD001 and FD003, that have a single operating condition, (Listou Ellefsen et al., 2019) achieved the best results. However, for subsets FD002 and FD004, that have 6 operating conditions, the CNN and MLP models from this work take the lead (in fact, all three models from this work outperform the others). Something similar happens for Table 4, but LSTM from this work takes the lead in FD002 and FD004 instead, losing for other algorithms in FD001 and FD003.

The better performance observed for the subsets FD002 and FD004 is largely explained by the use of operating condition-specific standardization, which makes the features somewhat invariant to operating conditions, making it easier for models to detect the degradation trends. In subsets FD001 and FD003, with a single operating condition, Eq. (2) is reduced to Eq. (1), and the models perform comparably, but not always better, to others from the literature.

Analyzing Table 5 it can be seen that the CNN got the best score on the test set for FD005. When comparing this result with the PHM Society 2008 Data Challenge Competition

Table 3. Test set RMSE comparison between the best results obtained in this work and the state of the art algorithms found in the literature for the TEDS datasets. Best results for each category highlighted in bold.

			FD001		FD002		FD003		FD004	
Publication	$r_{\max}$	Technique	Mean	STD	Mean	STD	Mean	STD	Mean	STD
This work	80	MLP	4.5	0.1	5.8	0.1	4.6	0.2	5.7	0.2
This work	80	CNN	4.3	0.1	5.6	0.2	5.0	0.2	6.4	0.2
This work	80	LSTM	4.9	0.3	6.0	0.3	5.0	0.3	6.1	0.2
This work	100	MLP	7.7	0.1	9.6	0.2	7.8	0.3	8.9	0.3
This work	100	CNN	8.3	0.4	9.5	0.2	8.3	0.3	9.6	0.2
This work	100	LSTM	8.7	0.5	10.2	0.6	8.3	0.4	9.7	0.3
This work	130	MLP	15.1	0.4	18.0	0.4	14.3	0.7	**16.6**	0.5
This work	130	CNN	15.0	0.5	**17.5**	0.7	14.8	0.8	17.4	0.9
This work	130	LSTM	16.5	0.3	18.1	0.9	15.9	1.0	17.2	1.4
(Ramasso, 2014)	135	RULCLIPPER	13.3	-	22.9	-	16.0	-	24.3	-
(Sateesh Babu et al., 2016)	130	CNN	18.5	-	30.3	-	19.8	-	29.2	-
(Zhang et al., 2017)	130	MODBNE	15.0	-	25.1	-	12.5	-	28.7	-
(Zheng et al., 2017)	130	LSTM	16.1	-	24.5	-	16.2	-	28.2	-
(Li et al., 2018)	125	CNN	12.6	0.2	22.4	0.3	12.6	0.1	23.3	0.4
(Listou Ellefsen et al., 2019)	115/135/ 125/135	RBM + LSTM	**12.6**	-	22.7	-	**12.1**	-	22.7	-

Table 4. Test set Score comparison between the best results obtained in this work and the state of the art algorithms found in the literature for the TEDS datasets. Best results for each category highlighted in bold.

			FD001		FD002		FD003		FD004	
Publication	$r_{\max}$	Technique	Mean	STD	Mean	STD	Mean	STD	Mean	STD
This work	80	MLP	57	7	574	75	81	5	193	15
This work	80	CNN	58	4	289	83	88	10	259	19
This work	80	LSTM	62	8	155	8	85	6	310	322
This work	100	MLP	124	10	780	174	232	12	562	42
This work	100	CNN	166	15	395	30	174	22	559	101
This work	100	LSTM	155	26	400	53	165	35	482	48
This work	130	MLP	411	54	1113	417	1091	195	2755	789
This work	130	CNN	369	64	1757	579	332	35	1678	143
This work	130	LSTM	444	79	**942**	155	718	383	**1487**	224
(Ramasso, 2014)	135	RULCLIPPER	**216**	-	2796	-	317	-	3132	-
(Sateesh Babu et al., 2016)	130	CNN	1286	-	13570	-	1596	-	7886	-
(Zhang et al., 2017)	130	MODBNE	334	-	5585	-	422	-	6558	-
(Zheng et al., 2017)	130	LSTM	338	-	4450	-	852	-	5550	-
(Li et al., 2018)	125	CNN	273	24	10412	544	284	27	12466	853
(Listou Ellefsen et al., 2019)	115/135/ 125/135	RBM + LSTM	231	-	3366	-	**251**	-	2840	-

Table 5. Comparison between the best results obtained for each of the different architectures and RUL limits for the PHM2008 dataset. RMSE and normalized score are taken from the validation set, since for FD005 the ground truth RUL is not disclosed. Score from test set also included for reference (evaluation only allowed once a day).

			Validation Normalized Score		Validation RMSE		Test Score
Publication	$r_{\max}$	Technique	Mean	STD	Mean	STD	
This work	80	MLP	0,82	0,02	7,30	0,16	-
This work	80	CNN	0,96	0,07	7,75	0,23	-
This work	80	LSTM	1,04	0,14	8,15	0,43	-
This work	100	MLP	1,77	0,06	10,44	0,25	-
This work	100	CNN	1,75	0,11	10,52	0,40	-
This work	100	LSTM	2,07	0,12	11,37	0,33	-
This work	130	MLP	5,13	0,46	**16,91**	0,25	1884
This work	130	CNN	**4,84**	0,24	17,39	0,39	**1314**
This work	130	LSTM	6,93	0,23	20,40	0,68	1505

rank, as described in (Ramasso & Saxena, 2014), we see that result would have reached 7th place.

5. CONCLUSION

Prognostics is a major enabler capability for maintenance of highly complex and safety critical systems such as the ones present in aircraft. The high associated costs with logistics and maintenance efforts in aeronautical industry are an encouraging factor for the application of prognostics and health management technologies, allowing for anticipated action planning which not only helps reduce costs but also improves system availability and overall safety.

In this work, a case study was performed on the application of data-driven prognostics methods based on neural networks. One of the significant selling points of data-driven methods is the fact that very little context specific knowledge is required about the inner workings of the system. Since high fidelity dynamic models are difficult to devise or in some cases even infeasible, this seems like a good compromise between performance, flexibility and applicability. Neural networks however do have many free parameters and high computational cost for training, which can be a disadvantage when compared to more specialized algorithms such as RULCLIPPER (Ramasso, 2014), which is almost parameter free.

The publicly available C-MAPSS datasets, provided by NASA, were used in this case study, and three different model architectures were evaluated: MLP, CNN and LSTM. The many design choices, exploratory analysis, preprocessing steps, and other relevant aspects of the implementation of these prognostics approaches were explored in search for the best resulting model.

A trade-off between prediction timeliness and accuracy was identified, where one may choose to decrease the model's prediction horizon, getting lower prediction errors in return. This improvement however may not necessarily result in more useful models or translate to other performance metrics.

A comparison with the state-of-the-art models (to the knowledge of the authors) for this problem was provided. The approaches explored in this work outperformed these models from literature in some of the data subsets with multiple operating conditions and achieved comparable results in the others. These results point to the conclusion that proper data cleaning and preprocessing based on solid exploratory analyses can lead to significant improvements when compared to directly feeding raw data into predictive models.

NOMENCLATURE

The nomenclature used in this paper is as follows:

$\boldsymbol{X}$	Multivariate time series (2D array)
$\boldsymbol{X}_{i:j,p:q}$	Array with time i to j and features p to q
$\boldsymbol{X}_{:,d}$	Array with all time steps, but only feature d
$\boldsymbol{X}_{t,:}$ or $\boldsymbol{X}_t$	Array with all features, but only time step t
$X_{t,d}$	Scalar observation of feature d at index t
$\boldsymbol{X}^{(u)}$	u-th 2D array
$\boldsymbol{y}$	Univariate time series (1D array)
y_t	Scalar observation at index t
$\boldsymbol{y}^{(u)}$	u-th 1D array
$\tilde{X}$ or $\tilde{\boldsymbol{X}}$	Normalized/standardized quantity
$\hat{y}$ or $\hat{\boldsymbol{y}}$	Estimated quantity
$\boldsymbol{X} \odot \boldsymbol{W}$	Hadamard (or element-wise) multiplication

REFERENCES

Ahmadzadeh, F., & Lundberg, J. (2013). Remaining useful life estimation: review. *International Journal of Systems Assurance Engineering and Management*, *5*(4), 461–474.

Guo, L., Li, N., Jia, F., Lei, Y., & Lin, J. (2017). A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, *240*, 98–109.

Javed, K., Gouriveau, R., & Zerhouni, N. (2017). State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels. *Mechanical Systems and Signal Processing*, *94*, 214–236.

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. In *Proceedings...* Ithaca, NW: Cornell University.

Li, X., Ding, Q., & Sun, J. Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*, 1–11.

Listou Ellefsen, A., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering and System Safety*, *183*, 240–251.

Liu, J., Saxena, A., Goebel, K., Saha, B., & Wang, W. (2010). An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. In *Proceedings...* (pp. 1–8). PHM Society.

Pecht, M., & Jaai, R. (2010). A prognostics and health management roadmap for information and electronics-rich systems. *Microelectronics Reliability*, *50*(3), 317–323.

Peel, L. (2008). Data driven prognostics using a Kalman filter ensemble of neural network models. In *Proceedings...* (pp. 1–6). Psicataway: IEEE.

Peng, Y., Xu, Y., Liu, D., & Peng, X. (2012). Sensor selection with grey correlation analysis for remaining useful life evaluation. In *Proceedings...* (pp. 1–10). PHM Society.

Ramasso, E. (2014). Investigating computational geometry for failure prognostics in presence of imprecise health indicator: results and comparisons on C-MAPPS datasets. In *Proceedings...* (Vol. 5, pp. 1–13). PHM Society.

Ramasso, E., & Saxena, A. (2014). Performance benchmarking and analysis of prognostic methods for CMAPSS datasets. *International Journal of Prognostics and Health Management*, *14*, 1–15.

Sateesh Babu, G., Zhao, P., & Xiao, L. (2016). Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. In S. Navathe, W. Wu, S. Shekhar, X. Du, W. X. Sean, & X. Hui (Eds.), *Database systems for advanced applications* (pp. 214–228). Cham: Springer. ((Lecture Notes in Computer Science, v. 9642))

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. In *Proceedings...* PHM Society. doi: 10.1109/PHM.2008.4711436

Saxena, A., & Goebel, K. (2008a). *Phm08 challenge data set.* Moffett Field, CA: NASA Ames Research Center, 2008. NASA Ames Prognostics Data Repository. Retrieved from `http://ti.arc.nasa.gov/project/prognostic-data-repository`

Saxena, A., & Goebel, K. (2008b). *Turbofan engine degradation simulation data set.* Moffett Field, CA: NASA Ames Research Center, 2008. NASA Ames Prognostics Data Repository. Retrieved from `http://ti.arc.nasa.gov/project/prognostic-data-repository`

Wang, T. (2010). *Trajectory Similarity Based Prediction for Remaining Useful Life Estimation* (Unpublished doctoral dissertation). University of Cincinnati.

Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for engineered systems. In *Proceedings...* (pp. 4–9). PHM Society.

Yang, F., Habibullah, M. S., Zhang, T., Xu, Z., Lim, P., & Nadarajan, S. (2016). Health index-based prognostics for remaining useful life predictions in electrical machines. *IEEE Transactions on Industrial Electronics*, *63*(4), 2633–2644.

Zhang, C., Lim, P., Qin, A. K., & Tan, K. C. (2017). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2306–2318.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *Proceedings...* (pp. 88–95). Psicataway: IEEE.

BIOGRAPHIES

Gabriel Duarte Pasa holds a B.S. degree in Control and Automation Engineering from Universidade Federal de Minas Gerais, Brazil (2015) and a MEng. degree in Aeronautical Engineering from Instituto Tecnológico de Aeronáutica, Brazil (2019). His research interests include neural networks and evolutionary algorithms and he has been working as an Avionics Systems Product Development Engineer at Embraer since 2017.

Ivo Paixão de Medeiros Computer Engineer (2008, Federal University of Pará), MSc. in Eletronics and Computer Engineering (2011, Aeronautics Institute of Technology) and Applied Computing D.Sc. (National Institute for Space Research). His research focuses on Integrated Vehicle Health Management and Prognostics and Health Monitoring. He has been working for Embraer since 2010, first at Flight Safety Dept. and since 2012, at Technology Development Dept.

Takashi Yoneyama is a Professor of Control Theory with the Electronic Engineering Department of the Instituto Tecnológico de Aeronáutica (ITA). He received his bachelor's degree (1975) and master's degree (1979) in Electronic Engineering from Instituto Tecnológico de Aeronáutica (ITA), Brazil, the M.D. degree in Medicine (1993) from Universidade de Taubaté (UNITAU), Brazil, and the Ph.D. degree in Electrical Engineering (1983) from the Imperial College London, U.K. (1983). He has more than 400 published papers, has written four books, and has supervised more than 100 Ph.D. and masters thesis. His research is concerned mainly with stochastic optimal control theory. He served as the President of the Brazilian Automatics Society in the period of 2004-2006.