CS230

Prognostic Health Management for Turbofan Engines

Aditya Gulati
adgulati@stanford.edu

Jeetsagar Ghorai
jghorai@stanford.edu

Abstract

Prognostic Health Management (PHM) is an active area of research and a multi-billion dollar industry in the field of reliability engineering. Complex sensor data exhibited by machines can be used to predict a bad omen (possible failure) beforehand, further saving downtime or loss of equipment and environment. We explore various deep learning solutions to model the spatio-temporal relationships exhibited by NASA Turbofans. Current approaches use CNN based models to predict Remaining Useful Life (RUL) of a system, we propose a novel CNN-LSTM architecture and explore the power of LSTMs to model sequential data. We further show the insights of deployment in terms of system tolerance statistics.

1 Introduction

Prognostic Health Management (*PHM*) is a unified framework for forecasting system health and reliability. Most systems of interest are composed of multiple components. Failure of a component in a system can result in adverse outcomes such as stoppage of operation, destruction of the system or loss of life. In most cases, the failure of a component results from the degradation of said component over the course of operation. Prognostic Health Management is concerned with forecasting potential failures of systems by monitoring the status of the components and the performance of the system.[1]

In most problems of interest, the data is available in the form of a time series of sensor readings. Given this time series data, the aim of PHM is to predict the Remaining Useful Life (*RUL*) of the system. Predicting the potential failure of a component allows the operator to plan for repair or replacement, mitigate downtime and ensure the safety of the equipment and the environment. Overestimating RUL leads to an unplanned failure, whereas underestimating RUL leads to under-utilization of the component.

The PHM problem which we have worked on in this paper is a regression problem. Given multivariate time-series sensor data we map it to the RUL at every given time-step. Such problem was usually solved by domain experts who are aware of the system dynamics, however with increasing complexity of system everyday it is almost impossible for human to model all complex non-linear relationships in the system. In this paper, we have introduced and explored several deep learning techniques to solve the problem.

2 Related work

PHM is an active area of research in reliability engineering and PHM techniques have been applied to a variety of systems such as hydraulic pumps[8], Lithium ion batteries[7], MOSFETs[6] etc. There are various statistical[10], signal processing[9], machine learning[11] and deep learning[4] methods in sensor data analytics. In paper[4] the authors use 1-D CNN and model it as a time-series regression problem. This paper acts as a baseline and starting point for our work. In DeepSense[5], the authors

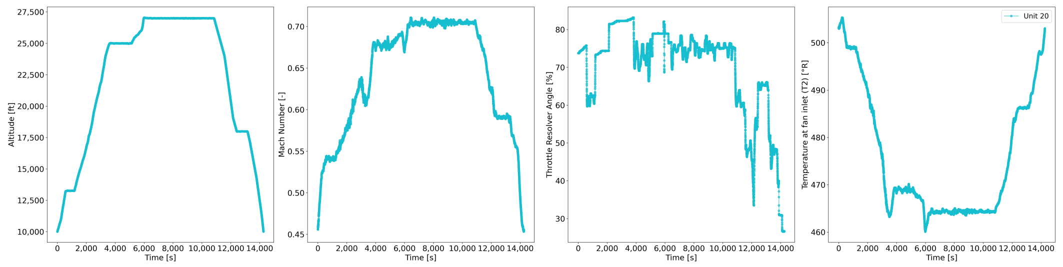

Figure 1: How the altitude, Mach number, throttle-resolver angle and temperature at fan inlet changes throughout a single flight of unit 20

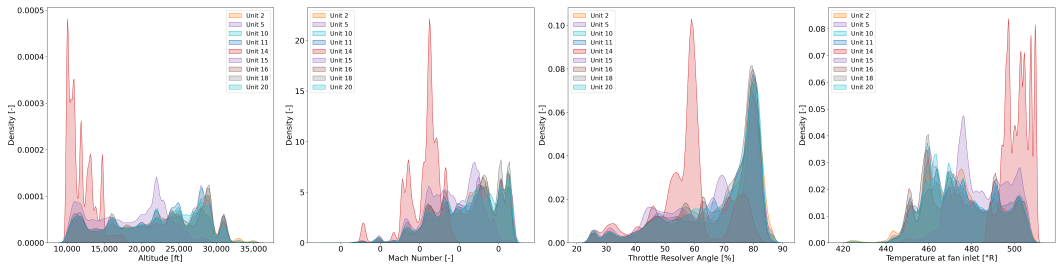

Figure 2: Kernel density estimations of altitude, Mach number, throttle resolver angle and temperature at fan inlet for 6 training and 3 test units. The flight characteristics of units 14 and 15 are different from those of the training units.

propose a novel CNN-LSTM architecture to model complex spatio-temporal relationships in time-series data. Although the paper is used as tracking methods, we take this as an inspiration for our project.

3 Dataset

We have used the Turbofan Engine Degradation Simulation Data Set-2 published by the Prognostics Center of Excellence at NASA. This dataset contains run-to-failure trajectories of a number of turbofan aircraft engines.[3]

The published repository contains multiple datasets. One representative dataset, `DS02`, consists of run-to-failure simulation data for nine engines. In this dataset, the operating conditions are described using 4 attributes (W). The model outputs the values of 14 measured physical properties (X_s), the readings from 14 virtual sensors (X_v) and 3 model efficiency parameters (θ). Together, there are 35 features at every time-step. In the dataset, the different engines are referred to as units. The units with $u = 2, 5, 10, 16, 18, 20$ are the six training units (training set) and the units with $u = 11, 14, 15$ are the three test units (test set). Some characteristics of the dataset are shown in Figure 4 and Figure 2.

It is worth noticing in Figure 2 that operating conditions for unit 14 (marked in red) significantly deviates from all the other engines, this makes our problem more challenging and demands a robust well-generalised solution. Table 1 describes the unit wise distribution of the whole dataset along with total number of cycles operated till end of life. One cycle implies a journey from take-off to landing, our RUL in dataset is in terms of cycles. We predict the remaining useful cycles under which the engine can operate at a given time.

4 Methods

Based on the available sensor attributes we divide our methods primarily into two different types of approaches. 1) Data-driven approach where we directly use the raw-sensor data and map it to RUL, 2) UKF based physics calibrated model. Both of the approaches are described below and in each approach we use CNN, CNN-LSTM and CNN-2StackedLSTM as deep learning model.

Unit	m_i	t_{EOL}
2	0.85M	75
5	1.03M	89
10	0.95M	82
16	0.77M	63
18	0.89M	71
20	0.77M	66
11	0.66M	59
14	0.16M	76
15	0.43M	67

Table 1: Unit-wise distribution, Unit 2,5,10,16,18,20 are in train set and 11,14,15 are in test set

.

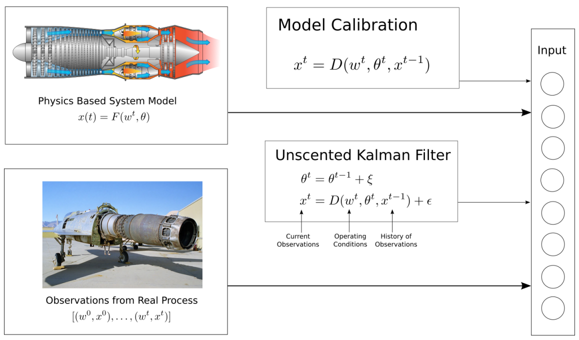
Physics Based System Model
$x(t) = F(w^t, \theta)$
Model Calibration
$x^t = D(w^t, \theta^t, x^{t-1})$
Input
Unscented Kalman Filter
$\theta^t = \theta^{t-1} + \xi$
$x^t = D(w^t, \theta^t, x^{t-1}) + \epsilon$
Current Observations
Operating Conditions
History of Observations
Observations from Real Process
$[(w^0, x^0), \ldots, (w^t, x^t)]$

Figure 3: The non-linear system dynamical physics model is first approximated as a FNN (trained offline), which is used as a system model in the UKF tracking algorithm, UKF further results in the physics enhanced input space which is merged with the simulation input space resulting $[w, x_s, \hat{x_s}, \hat{x_v}, \hat{\theta}]$ as the final physics-enhanced input. This further serves as our input for the deep learning model.

4.1 Data-driven model

In this approach we learn a mapping from enhanced input space $[w, x_s, x_v, \theta]$ to RUL target Y using a particular neural network architecture. The enhanced input space in the dataset is obtained after using a simulation software N-CMAPPS which deviates from the real physics process of the turbofan engine. To mitigate this reality gap we present a calibration based approach in next subsection.

4.2 UKF calibrated physics-based model

The sensor readings x_s, x_v and model health parameters θ are obtained from a simulation software. This sensor consists of noise and is much deviated from the real world process. To counter this we use an Unscented Kalman Filter (UKF) as a tracking algorithm and induce the system physics. UKF tracking algorithm requires the current sensor readings and physics model as input along with some hyper-parameters to drive the iterative process. The above tracking algorithm provides us with a more accurate and correlated (physics enhanced) inputs $[\hat{x_s}, \hat{x_v}, \hat{\theta}]$ which are merged with operating conditions and sensor inputs $[w, x_s]$. The resultant physics enhanced input space is $[w, x_s, \hat{x_s}, \hat{x_v}, \hat{\theta}]$ which is mapped to target RUL Y using the neural network architecture. Detailed flow of this physics-based modelling is given in description of Figure 3.

4.3 Model Architectures

The data is first divided into 50xC 2D-vector segments for 50 timestamps (with C sensor attributes), this input is fed into a 1D-CNN[4], further mapped to RUL. The loss function used here is Mean Squared Error. This approach acts as baseline result. As the data fed is time-series we the final layer of base-cnn models spatial sequential relationships. Recurrent Neural Networks (*RNN*), perform

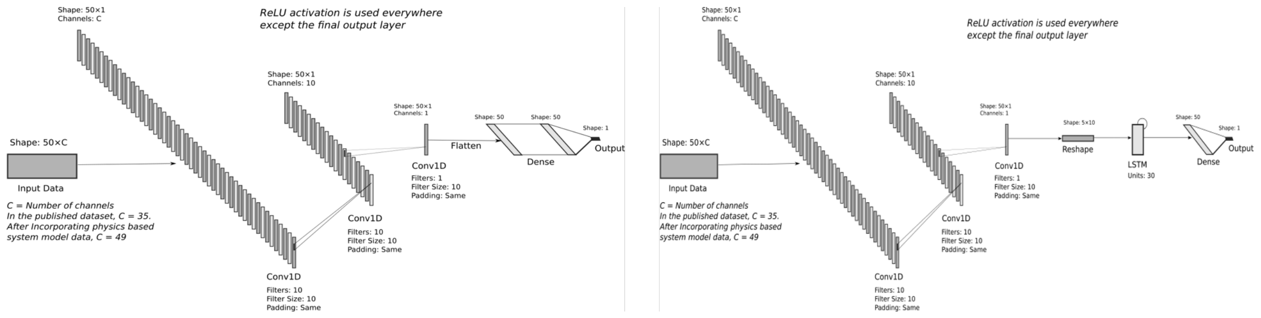

Figure 4: (L to R) 1) Baseline 1-D CNN with Dense layer at the end. 2) CNN-LSTM Architecture modelling the temporal relationships at the end of network, stacinkg of another LSTM here would make it CNN-2StackedLSTM

exceptionally well at prediction tasks involving temporal sequences. The dataset under consideration consists of time series data of sensor readings. That is why, we decided to investigate the potential of RNN networks for predicting the RUL. In our experiments, we used LSTM networks instead of RNN. More details of architecture are given in description of Figure 4. In next section we discuss the performed hyper-parameter tuning and results of all networks under all models.

5 Experiments

We have explored three types of architectures BaseCNN, CNN-LSTM and CNN-2StackedLSTM. Under each architecture we perform our experiments on both types of methods (data-driven and UKF physics based method). We train our neural networks for 60 epochs with different batch sizes (256,512,1024). The other architecture based hyper-parameters are filter kernel size (5,7,10), Dropout (0,0.2), Number of filters in each layer (5,10,20), Batch-Normalisation (Yes/No) and sequence length of layer fed into LSTM (5x10, 10x5).

The most important hyper-parameter in our experiments is choosing the filter size, a lower size can lead to under-fitting and a higher number can lead to over-fitting on the training set. Hence we first tune on this hyper-parameter. The sequence length of input to LSTM is also an important parameter as it actually defines the temporal nature of the data. Some of the results from above tuning are tabulated and explained in next section.

6 Results

The evaluation metric which we have used in our experiments are RMSE and NASA's scoring function (s) (between actual and predicted RULs).

$$s = \sum_{j=1}^{m_*} exp(\alpha|\Delta^{(j)}|)$$

$$RMSE = \sqrt{\frac{1}{m_*}\sum_{j=1}^{m_*}(\Delta^{(j)})^2}$$

where $\Delta^{(j)}$ is difference between predicted and actual RUL, m_* is length of test-dataset. In scoring function $\alpha = \frac{1}{13}$ when RUL is under-estimated and $\alpha = \frac{1}{10}$ when RUL is over-estimated, hence penalising it more when RUL is over-estimated which is more costly.

The above metrics are calculated and tabulated in Table 2 for different set of experiments. We observe the best performance after using CNN-LSTM architecture with physics augmentation with 10 filters, 5 kernel size and 5x10 sequences in LSTM.

We analyze the unit wise results obtained by our best performance model and provide deployment insights. In figure 5 we can see how our best model fits well on 2 test units (11 and 15), and remains as underestimate on training set. In figure 6 we take mean of predicted RUL per actual cycle and set

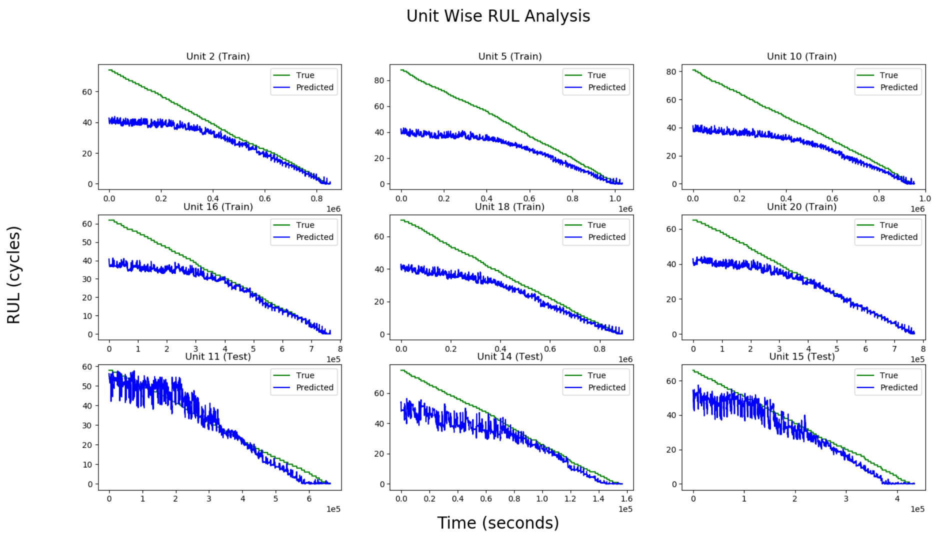

Figure 5: Unit wise predicted and true RUL for best (highlighted) model.

a deploy-able threshold of 8, we can observe unit 11 and 15 remains in deploy-able time for long duration and unit 8 for considerable amount of time. More details are given in the description of figure 6.

Model	RMSE	NASA fn (s * 10^6)
BestCNN	7.662	2.15
BestCNN-Physics	6.9274	2.16
BestCNN-LSTM	7.4127	2.21
BestCNN-LSTM-Physics	**6.6639**	**2.01**
BestCNN-2LSTM	7.7764	2.23
BestCNN-2LSTM-Physics	6.81459	2.05
CNN-Kernel10	8.4524	2.85
CNN-Filters20	7.9657	2.34
CNN-Filters5	9.8745	2.56
CNN-LSTM10x5	7.9564	2.24

Table 2: Evaluation of different models on test set

.

7 Conclusion and Future Work

We have explored the power of LSTMs to model temporal relationships in sequential data and showcase that it performs better than only CNN. Currently we have modeled our problem only on one fleet of engines in our dataset (DS02), exploring the power of domain adaptation or transfer learning to utilise the learnings on another fleet can be a good future research work.

8 Contributions

Both the authors have contributed equally as a team to this project. Specifically, Aditya worked on formulating/designing the problem, enabling the physics based model and compiled everything in a Python-UI to show a visualisation of deployment (shown in video) further enabling it to be an end-end AI product. Jeetsagar worked on preprocessing the data, preparing the data pipeline for both tensorflow and torch approaches along-with exploring the scope of applicability of LSTMs for our problem. We thank our CS230 course mentor Huizi Mao for guiding us throughout this project with his encouraging feedback.

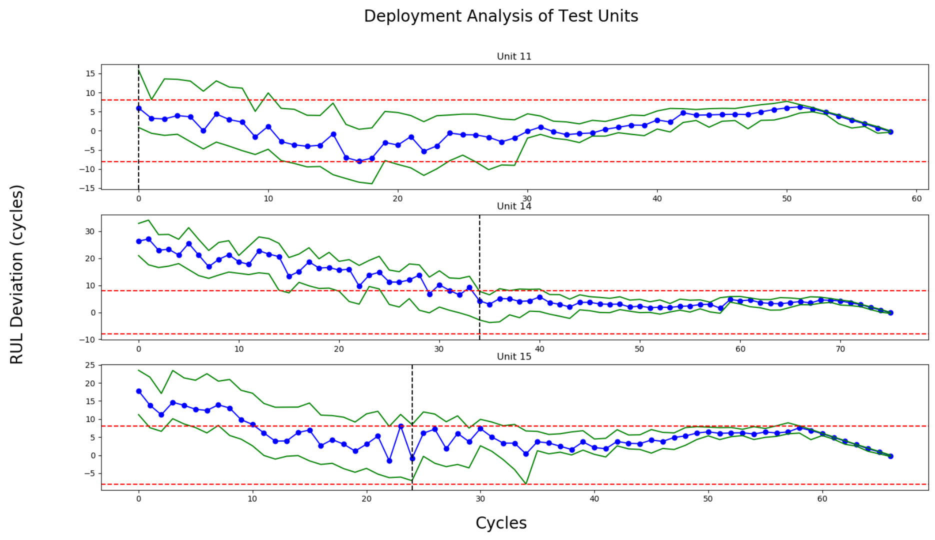

Figure 6: Deployment Statistics for best performing model. Horizontal red line defines the acceptable deviation range, blue line represents the actual deviation, green line represents horizon of deviation and black line represents the maximum cycle after which we are always in deploy able range.

References

[1] Kwok L. Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, Wenbin Wang, "Prognostics and Health Management: A Review on Data Driven Approaches", Mathematical Problems in Engineering, vol. 2015, Article ID 793161, 17 pages, 2015. https://doi.org/10.1155/2015/793161

[2] Biggio Luca, Kastanis Iason, "Prognostics and Health Management of Industrial Assets: Current Progress and Road Ahead", Frontiers in Artificial Intelligence,VOL 3, 2020, 88 pages, https://www.frontiersin.org/article/10.3389/frai.2020.578613, 10.3389/frai.2020.578613, 2624-8212

[3] M. Chao, C.Kulkarni, K. Goebel and O. Fink (2021). "Aircraft Engine Run-to-Failure Dataset under real flight conditions", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA

[4] Chao, Manuel Arias et al. "Fusing Physics-based and Deep Learning Models for Prognostics." ArXiv abs/2003.00732 (2020)

[5] Shuochao Yao and Shaohan Hu and Yiran Zhao and Aston Zhang and Tarek Abdelzaher, "DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing", ArXiv abs/1611.01942

[6]Renwick J., and Kulkarni C. and Celaya J., Analysis of Electrolytic Capacitor Degradation under Electrical Overstress for Prognostic Studies, Annual Conference of the Prognostics and Health Management Society, 2015

[7]Liu, Datong and Luo, Yue and Peng, Yu and Peng, Xiyuan and Pecht, Michael, Lithium-ion battery remaining useful life estimation based on nonlinear ar model combined with degradation feature, Annual Conference of the Prognostics and Health Management Society 2012, 24–27, 2012

[8] Goebel, Kai Frank,Management of uncertainty in sensor validation, sensor fusion, and diagnosis of mechanical systems using soft computing techniques, Thesis, University of California, Berkeley, 1996

[9] Wang, Fengtao and Zhang, Yangyang and Zhang, Bin and Su, Wensheng, Application of Wavelet Packet Sample Entropy in the Forecast of Rolling Element Bearing Fault Trend, Multimedia and Signal Processing (CMSP), 2011 International Conference on, 12–16, 2011

[10] Olivares, Benjamín E and Munoz, Cerda and Orchard, Marcos E and Silva, Jorge F, Particle-filtering-based prognosis framework for energy storage devices with a statistical characterization of state-of-health regeneration phenomena,Instrumentation and Measurement, IEEE Transactions on, Vol. 62 No. 2, 364–376, 2013

[11]Contribution of belief functions to hidden markov models with an application to fault diagnosis., Ramasso, Emmanuel, Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing., 1–6, 2009