# LSTM-Based Approach for Remaining Useful Life Prediction of Air Craft Engines

Rupali Upadhyay[1], Hemant Amhia[2]
[1,2]EE Department, Jabalpur Engineering College, Jabalpur

**Abstract:** Working machinery has a limited lifespan and occasionally breaks down due to obsolete functioning. A maintenance strategy must be used on the scheduled machinery system in order to prevent the worst scenario (failure) and learn more about the machine's condition. The ideal maintenance technique is predictive maintenance. Predictive maintenance, whose major goal is to forecast hardware component failures by continually monitoring their status, emphasizes the importance of this requirement so that maintenance operations can be planned far in advance. These observations, which span the lifecycle of the corresponding components, are produced by monitoring systems and often take the form of time series and event logs. The fundamental difficulty of data-driven predictive maintenance is analysing this history of observations in order to create prediction models. Machine learning has become widely used in this direction as a result of its ability to extract knowledge from a range of data sources with the least amount of human involvement. This work aims to investigate and deal with difficult issues in aviation connected to foreseeing component breakdowns on board. Scalability is a crucial component of every suggested strategy due to the vast volume of data related to the operation of airplanes.

**Introduction:**

The core of an airplane is its turbofan engines, and study of their health condition is crucial for assessing aircraft, ensuring safe operation, and developing maintenance plans. However, The main metric used to evaluate the condition of turbofan engines is the RUL. Turbofan engines' intelligence and productivity have recently increased significantly as a result of the quick development of machine learning and deep learning. In addition to preventing safety hazards brought on by delayed maintenance, Reducing the excessive expense of unneeded maintenance through accurate estimation of remaining usable life. The RUL prediction of turbofan engines, however, presents significant difficulties in feature extraction and prediction accuracy because of the multiple measuring locations, complicated working conditions, and large amounts of data. First, the usefulness of features is directly influenced by selecting a model or feature extraction method. Second, the health indicators (HIs) demonstrating the degeneration of turbofan engines are used to gauge the forecast's accuracy. Machine learning-based analytics could be useful for predicting turbofan core size.

The use of machine learning analytics for turbofan life prediction is the main emphasis of this work. The “heart” of the airplane is its turbofan engine, a piece of thermal technology that is incredibly intricate and precise. The turbofan engine is a contributing factor in about 60% of all airplane failures. The engine first functions normally before eventually developing a problem. The project’s goal is to put different Predictive Maintenance techniques into practice and evaluate each technique’s effectiveness. The objective of this work is to implement distinct

Predictive Maintenance methods and assess the performances of each method. The methods are broadly classified in two categories. First, is the Classification Method, where The goal is to forecast the engine's breakdown in the following days. by implementing binary classification models Second, is the Regression Method, where the goal is to implement a prediction of the engine's remaining useful life an exponential degradation model and long-short term memory (LSTM) deep learning model. The project mainly targets an engine manufacturer for aircraft gas turbine engines. The users are assumed to be from the manufacturing team of these aircraft engine corporations. A software component will be added to every engine sold that is now on the market to read the engine's sensor readings, use a model to anticipate the RUL, and respond when the RUL falls too low by triggering maintenance. The use case of this model is to describe a failure and give an early warning to generate a data label, ask the maintenance personnel. The model will then be routinely re-trained to enhance prediction quality over time using completed data series including this label. In [1], The long-term forecast of machine health conditions is presented in this work using a competitive learning-based methodology. Specifically, Rolling bearing vibration data is pre-processed using continuous wavelet transform. with defects (CWT). Then, candidate inputs to an RNN made up of statistical parameters generated from both the raw and the pre-processed data are used. In [2], The author of this paper suggests a RUL estimate using the Long Short-Term Memory (LSTM) technique that can completely utilize sensor sequence information and uncover occult patterns in sensor data across a range of operating conditions, malfunction, and degradation models. In [3], By building a 2-dimensional (2D) DCNN from the sensor signals' time series, The RUL of aviation engines could be predicted. DCNN has fewer parameters than other deep learning algorithms since it uses weight sharing technology and has better feature extraction capabilities. However, the DCNN only learns these nonlinear combination characteristics in a straightforward manner. Consequently, DCNN's capacity to look for the global optimum is constrained. In [4], This study presents a framework for estimating remaining usable life. The framework is divided into two sections: assessment of remaining usable life and creation of health features. Consequently, a new method for developing health features is suggested utilising a binary SVM classifier, which additionally generates defect detection as a separate feature then, using a well-known Weibull function, the remaining useful life is calculated. The Weibull parameters are discovered using a weighted least squares method. In [5], This paper focused developing a stacking ensemble of gradient-boosted trees and feed-forward neural networks for the RUL prediction challenge. This kind of technique directly uses the monitoring data collected at each time point as input to generate an independent prediction. The strategy, however, might lose the data correlation information for historical monitoring series that depend on time. In [6], For RUL prediction in this paper, the author utilised a stacked Sparse Autoencoder (SAE). Based on the grid search methodology, the hyperparameters of the SSAE were established. One of the approach for RUL prediction is combination of auto encoder and DBN[7]. Some traditional machine learning approaches are useful in RUL prediction[8]. One of author explains the importance of RUL prediction in industry 4.0 scenario[10]. The main contributions of this work are:

1. By hyperparameter tunning we found best fit combination of parameters.
2. In this work we explore the LSTM in deep by performing experiments with some changes in model.

## Data and Evaluation Parameters

### Data-Set

This dataset is an open-source data freely available on NASA’s data repository. This data basically a Run-to-Failure simulation data from turbo fan jet engines this data is also known as C-MAPS data.

Predicting the remaining usable life (RUL) of each engine in the test dataset is the ultimate purpose of this dataset. The RUL is the number of flights the engine can still perform after the final test data point. Each and every turbo engine starts off in this simulation with a variable degree of beginning conditions.

This data set consists of a number of multivariate time series. Training and test data are further separated from the data set. The data can be regarded as coming from the same type group of engines because each time series is specifically captured from a separate engine. This data consists of 21 sensors and three operational settings that have a big impact on engine performance. The data also contains these settings. Sensor noise has polluted the data.

In our work we have considered only the FD001 dataset with Conditions: ONE (Sea Level) and Fault Modes: ONE (HPC Degradation).

**Data Analysis**

The Data Schema of the used data of this paper is shown in Table 1. For this dataset, we have done some data analysis and all results are discussed below:
**a**. Data analysis is done on Train Data set as it has 20,631 rows and 27 columns including 21 sensor readings and 3 operational settings for each engine id.
**b**. After visualization for operational settings 1, 2, and 3 where operational settings 1 and 2 do not show a clear signal of failure, they might be important. Op set 3 remains the same throughout the engine's life. Therefore, we can drop this feature.
**c**. Similarly, after visualizing all the sensor values from sensor 1 to sensor 21 for each engine id, some sensors remain the same throughout the engine cycles.
**d**. Sensors 1,5,10,16,18,19 do not add value and we can remove them from analysis, to reduce the complexity of our model. In addition, sensor 6 remains constant as well with some minor fluctuations hence, we can remove it as well.
**e**. After plotting a heatmap figure 1, we observed that sensor 9 and sensor 14 were highly correlated with each other. Since both sensors are highly correlated and sensor 9 is correlated with RUL, we can drop sensor 14.

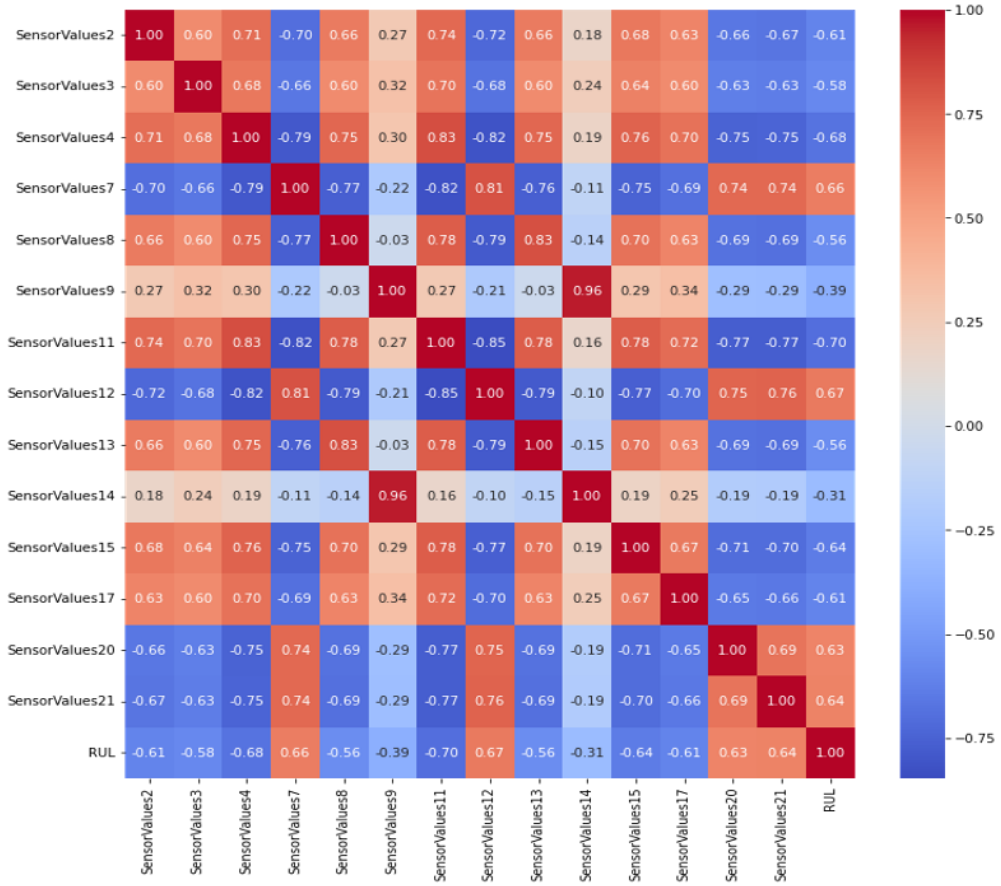


***Figure 1*** *Correlation Heatmap for all sensors*

**Evaluation Parameter**

The performance in this study is assessed using the root mean square error (RMSE) measure.

RMSE: It is the average of the squared discrepancies between the values that were predicted and the actual values.

$$\text{RMSE} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2}$$

**System Configuration:-**

All experiments are performed with system configuration shown below.

Intel i5, 8GB memory, 64bit, window 10, GPU: NVIDIA GeForce MX230, Python version 3.6 is used.

**Table 1** Dataset Schema

| Index | Column names | Data Descriptions |
|---|---|---|
| 1. | engine_number | Unit Number |
| 2. | CountOfCycles | Time, in cycles |
| 3. | op_setting1 | Operational Setting 1 |
| 4. | op_setting2 | Operational Setting 2 |
| 5. | op_setting3 | Operational Setting 3 |
| 6. | SensorValues1 | Sensor measurement 1 |
| 7. | SensorValues2 | Sensor measurement 2 |
| 8. | …… | ……. |
| 26. | SensorValues21 | Sensor measurement 21 |

**Methodology**

**LSTM**

The input order of data plays a crucial role in the predictive health monitoring algorithms. The goal is to pinpoint how factors change over time and how it affects RUL estimate. As a result, the inputs default format is sequential data. Examples of this data include pressure time series data or even variations in temperature over time. LSTM is most widely preferred for sequential data. LSTMs are purposefully developed to address the problem of long-term reliance. Instead of training to learn, LSTMs fundamentally have a natural ability to recall knowledge for long periods of time. As a result, we have decided to use LSTM.

The key feature of LSTMs is their ability to maintain long-term dependencies in the data by using a memory cell with three interacting gates:

**Forget gate:** Determines what information to forget from the previous time step's cell state. It takes input from the previous hidden state (h(t-1)) and the current input (x(t)), and outputs a forget vector (f(t)) with values between 0 and 1 for each element of the cell state. A value of 0 indicates "completely forget," while a value of 1 indicates "completely retain."

**Input gate:** Determines what new information to store in the cell state. It also takes input from the previous hidden state (h(t-1)) and the current input (x(t)) and outputs an input vector (i(t)) with values between 0 and 1 for each element of the cell state. A value of 0 indicates "do not store," while a value of 1 indicates "store completely."

**Output gate:** Determines the information to output from the current cell state. It takes input from the previous hidden state (h(t-1)) and the current input (x(t)), as well as the modified cell state from the forget and input gates. It outputs an output vector (o(t)) that scales the updated cell state to produce the current hidden state (h(t)).

The LSTM makes use of a cell that manages the input-output data error margins by storing memories of prior states. This makes it possible for a memory cell to retain its data for a considerable amount of time, which makes it easier to learn about long-term relationships that may affect RUL forecasting.

**Steps involved in this method:-**

Here we discussed steps involved in the LSTM method:
1. Data Pre-processing is applied on input dataset.
2. Dropping Sensors which do not affect the RUL.
3. Transforming train and test data into NumPy array including the target labels.
4. For preparing train and test data we considered sequence length 50.
5. To prepare test data we have additionally considered mask value zero.
6. Model is created using two layers with dropout layers as shown below figure 2.
7. A simple architecture of LSTM units is trained using RMSProp optimizer and Mean Squared Loss function for 100 epochs.
8. Next, we look at the performance on test data. We also plot the loss and compare the results on train and test data set.

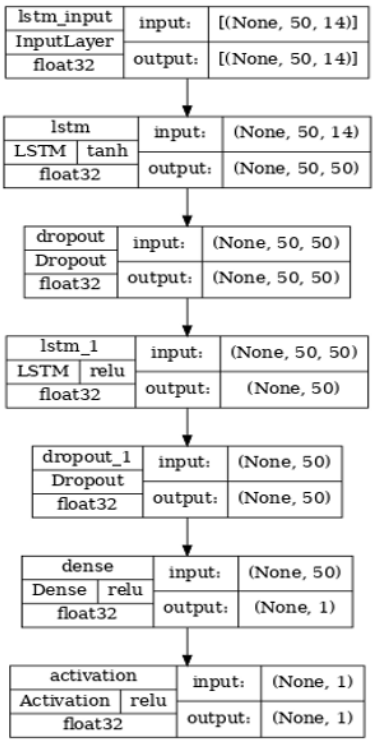


***Figure 2*** *LSTM Model*

In this case, a two-layer, dropout network is constructed. A second LSTM layer with 50 units is added after the first LSTM layer, which has 100 units in total, one for each input sequence, In order to prevent overfitting, we have additionally applied dropout to each LSTM layer. The prediction criteria is met by the ReLu activation used in the last dense output layer. In above figure 2 we can see LSTM model layers. Tanh and ReLu activation function is used. ReLu activation function introduces a non-linearity in the model it is responsible to reduce vanishing gradient issue. In figure 3 we can observe the ground and prediction plot for this method we can conclude from that difference between predicted value and the actual value is high. We have to improve train and test results by modifying some aspects.

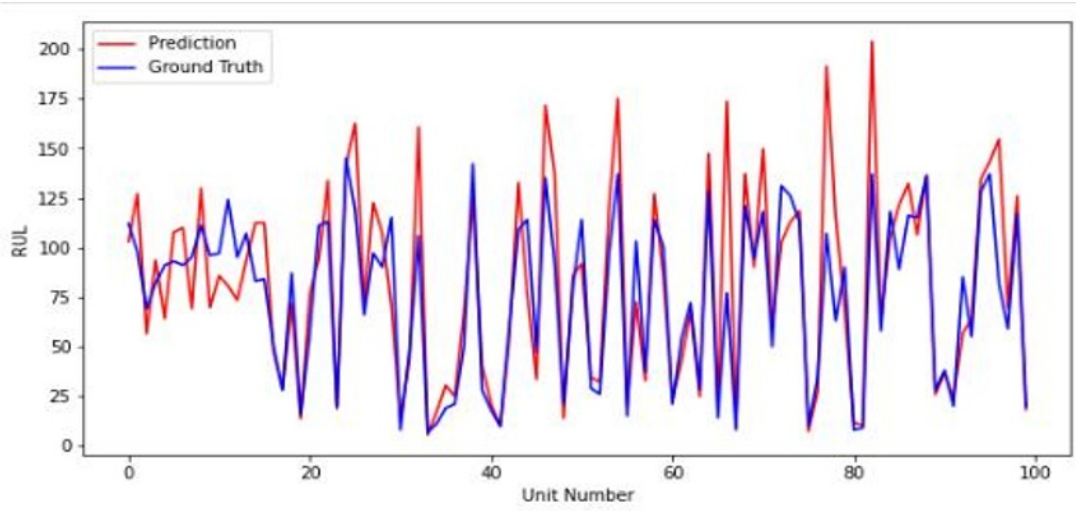


***Figure 3*** *Ground truth and prediction plot for LSTM*

**Test RMSE: 25.121725144160582**
**Train RMSE: 19.437863459976043**

**LSTM_EWMA**

As we can see in the above method there is huge difference between predicted and actual train and test results, so we are using exponential weighted moving average with the LSTM and let see what will happen then.

In figure 4 LSTM_EWMA model is shown, this model is created by modifying previous model. We developed a simple mechanism to rate our models. The two metrics I select to use are explained variance (or R2 score), which demonstrates how much of our dependent variable can be explained by the independent variables we use, and root mean squared error (RMSE), which displays the typical number of cycles the forecasts are incorrect.

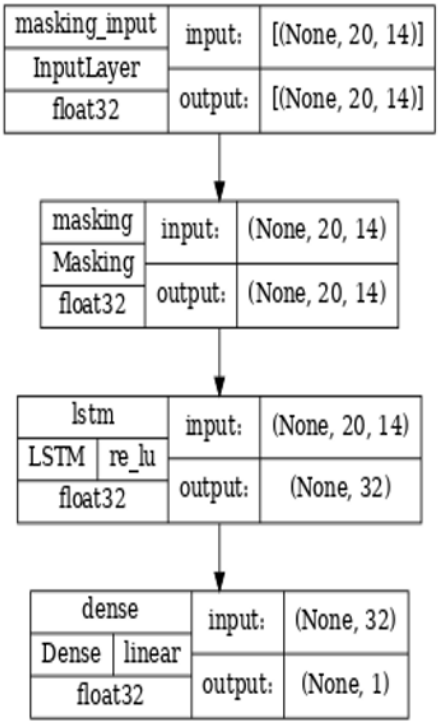


***Figure 4*** *LSTM_EWMA Model*

For smoothing, we've utilized an exponential weighted moving average. This smoothing feature works wonders. Essentially, it uses the current value and the previous filtered value to calculate the filtered value.

$$\sim Y_t = \alpha * Y_t + (1 - \alpha) * \sim Y_{t-1}$$

where $\sim Y_t$ is the exponential moving average at time t. Here alpha varies between 0 and 1, alpha is known as smoothing factor. $Y_{t-1}$The filtered datapoint will contain 40% of the value at Yt and 60% of the (previously filtered) value at Yt-1 when alpha is equal to 0.4. Therefore, lower alpha values will result in a strong smoothing effect. The result of predicted and ground truths are plotted in figure 5. We can see in the plot that predicted values are improved in this method as compared to previous method. But these are not very satisfied results we have to improve these results.

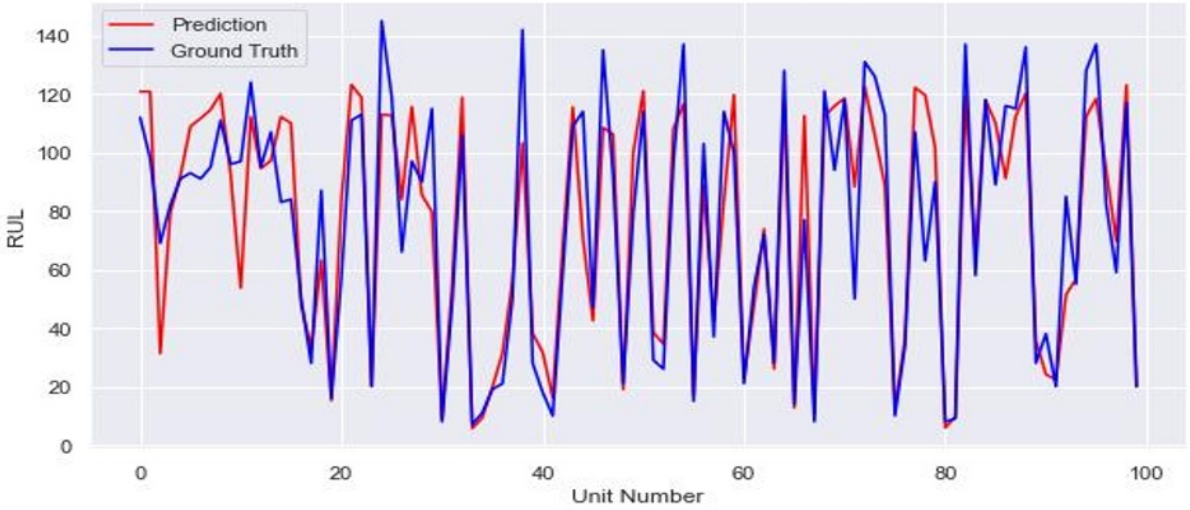


***Figure 5*** *Ground truth and prediction plot for LSTM_EWMA*

**Test RMSE: 17.658590116615787**
**Train RMSE: 13.371212005615234**

**Proposed Method**

In the previous two models results are not satisfying much we have to improve these results by modifying the previous method Now we are proposing our method LSTM_EWMA with hyperparameter tuning. Here we used LSTM_EWMA model as shown in figure 4 and apply hyperparameter tunning.

Here we are proposing a method for better results, LSTM_EWMA with Hyper-Parameter Tuning, Hyperparameter tuning is a way of getting the most optimal values of hyperparameters of a machine learning model. In this method, we apply hyperparameter tuning on the previous method to find hyperparameters of the LSTM_EWMA model.

In figure 6 flow diagram of the proposed model is shown. The deep model's hyperparameters significantly affect the outcomes. Although manual search and grid search are feasible in the context of this article, we employ an efficient and simple random search algorithm instead because applying manual search or grid search to fresh datasets is a bad idea.

Hyperparameters did not change during the training process so these are very crucial to achieve the best performance of your model.

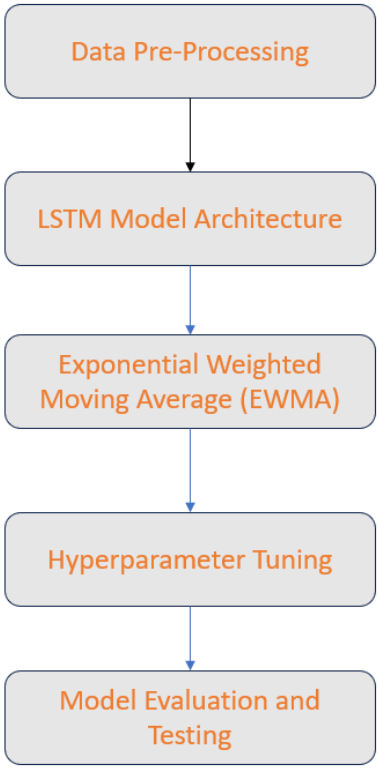


***Figure 6*** *Flow Diagram of Proposed Model*

In this experiment, we are planning to tune these hyperparameters: alpha, sequence length, epochs, nodes, dropout, activation function, batch size, and sensors.

**Steps Involved in this method:-**

- In this model, lower alpha's perform better, hence, we remove a few high ones to reduce sample space.
- Lowest dropout=0.1 is considered, Since there will be no dropouts, training outcomes will be greater but generalization will be worse.
- ReLu performed noticeably worse in early tests, so removed it.
- After preparing data and creating the model, we run the model for 15 iterations with the above shown parameters.
- Below are the iterations for the created model and the best parameters from those iterations are chosen to run the model again and results are derived.

In the figure 7 we can see the optimal values of the hyperparameters we will use these values of the hyperparameters to train the model again test the model again and find the results for proposed model. From the experiment, we found optimal values of the hyperparameters. We can see the values of hyperparameters in a highlighted row in Figure. 7.

After getting an optimal value of hyperparameters we have to use these values for training and testing the model and get results. We can see that after hyperparameter tuning results are improved. We can observe these results after hyperparameter tunning in table 2.

| | MSE | std_MSE | alpha | epochs | nodes | dropout | activation | batch_size | sequence_length | sensor_length |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1453.154826 | 22.16438 | 0.4 | 5 | [32] | 0.3 | tanh | 32 | 30 | 10 |
| **1** | 2650.680908 | 46.747098 | 0.01 | 5 | [256] | 0.1 | sigmoid | 256 | 30 | 14 |
| **2** | 3389.558675 | 72.247219 | 0.3 | 15 | [32, 64] | 0.1 | sigmoid | 256 | 15 | 10 |
| **3** | 324.659429 | 151.238837 | 0.01 | 15 | [128, 256] | 0.2 | tanh | 64 | 30 | 14 |
| **4** | 439.295013 | 54.187579 | 0.3 | 10 | [64, 128] | 0.4 | sigmoid | 32 | 10 | 10 |
| **5** | 192.03362 | 59.719933 | 0.4 | 15 | [64, 128] | 0.4 | sigmoid | 64 | 30 | 14 |
| **6** | 310.48112 | 36.687726 | 0.4 | 10 | [32] | 0.3 | tanh | 32 | 25 | 14 |
| **7** | 274.182281 | 30.291224 | 0.1 | 15 | [256] | 0.1 | tanh | 256 | 35 | 10 |
| **8** | 149.186615 | 33.642387 | 0.1 | 20 | [256] | 0.3 | sigmoid | 64 | 35 | 10 |
| **9** | 318.642802 | 70.457091 | 0.1 | 20 | [256] | 0.3 | sigmoid | 256 | 25 | 14 |
| **10** | 471.36617 | 58.723505 | 0.3 | 10 | [128] | 0.3 | tanh | 128 | 25 | 10 |
| **11** | 1711.674845 | 23.279815 | 0.01 | 5 | [256] | 0.2 | tanh | 128 | 35 | 10 |
| **12** | 2648.319417 | 77.231804 | 0.4 | 20 | [64] | 0.4 | sigmoid | 256 | 15 | 14 |
| **13** | 4698.539225 | 87.537954 | 0.05 | 15 | [32] | 0.1 | sigmoid | 256 | 15 | 10 |
| **14** | 352.364339 | 43.313729 | 0.6 | 15 | [256] | 0.2 | tanh | 128 | 15 | 10 |

***Figure 7*** *Hyperparameter Tuning Results*

**Test RMSE: 14.787540710863125**
**Train RMSE: 11.755559921264648**

**Table 2:** Results after getting optimal values of hyperparameters

| **Optimal values of hyperparameters** | **Train RMSE** | **Test RMSE** |
|---|---|---|
| **alpha** = 0.1<br>**epochs** = 20<br>**nodes_per_layer** = [256]<br>**dropout** = 0.3<br>**activation** = 'sigmoid'<br>**batch_size** = 64<br>**sequence_len** = 35<br>**sensor_len**=10 | **11.7555** | **14.7875** |

**Results discussion**

Different experiments were performed on the same data with improvements in methods. The ultimate goal of these experiments is to minimize the error factor in between predicted and actual values. After observing all experiments, we can say that while working on FD001 Dataset our proposed method gave better results in comparison to other experiments and other existing methods.

***Table 3*** *Compare different experiments*

| | **RMSE on FD001 Dataset** | |
|---|---|---|
| | **Train** | **Test** |
| **LSTM** | 19.43 | 25.12 |
| **LSTM_EWMA** | 13.37 | 17.65 |
| **Proposed Method** | **11.76** | **14.79** |

After finding the root mean squared error for train and test data from three experiments. We can see that there is a huge difference between the test RMSE of all three models for dataset FD001 The proposed method performs superior than others in terms of effectiveness. In table 3 we can see that proposed method is performing better than above two models.

***Table 4*** *Comparison of RUL Prediction between different existing methods*

| **Methods** | **Test RMSE on FD001 Dataset** |
|---|---|
| **1. Echo State Network[11]** | 63.45 |
| **2. Support Vector Machine [12]** | 29.82 |
| **3. Support Vector Regression [13]** | 20.96 |
| **4. Convolutional Neural Network [13]** | 18.44 |
| **5. Soft Computing method [14]** | 16.67 |
| **6. Deep LSTM [15]** | 16.14 |
| **7. Time Window-Based Neural Network [16]** | 15.15 |
| **8. The Proposed Method** | **14.79** |

In Table 4 we compared the Test RMSE of the proposed method to other existing methods we can see here that the proposed method is performing well for the FD001 dataset than other methods.

## Conclusion

Compare to the results of traditional machine learning methods on the FD001 dataset of C-MAPS our proposed method is giving better results we can observe that in table 4. As per the table 4 we can see that deep LSTM also have good results and time window based NN also have lower RMSE than other methods in RUL prediction.

For future work there is a lot of scope we will have to optimize the model to minimize training time, error, and computational complexity. The proposed method can be useful for the estimation of remaining useful life of aircraft engines with different operating conditions. RUL prediction field is complex and challenging so this problem require more study and research, we have to explore this area.

## References

1. Malhi, A., Yan, R., Gao, R.X.: Prognosis of defect propagation based on recurrent neural networks. IEEE Trans Instrum Meas 60(3), 703–711 (2011).
2. S. Zheng, K. Ristovski, A. K. Farahat, et al., Long short-term memory network for remaining useful life estimation, in Proc. ICPHM, Dallas, TX, USA, 2017, pp. 88–95.
3. X. Li, Q. Ding, and J. Q. Sun, 2018. Remaining useful life estimation in prognostics using deep convolutional neural networks. Reliability Engineering & System Safety, 172, pp.1-11.
4. C. Louen, S. X. Ding, C. Kandler. A new framework for remaining use-ful life estimation using the Support Vector Machine classifier. Conference on Control and Fault-Tolerant Systems, 2013, pp. 228-233.

5. S. K. Singh, S. Kumar, J. P. Dwivedi. A novel soft computing method for engine RUL prediction. Multimedia Tools and Applications, 2017, pp. 1-23.
6. J. Ma, H. Su, W. L. Zhao, and B. Liu, 2018. Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning. Complexity, 2018.
7. Huthaifa AL-Khazraji1, Ahmed R. Nasser1, Ahmed M. Hasan1, Ammar K. Al Mhdawi2, Hamed ,Al-Raweshidy3, Amjad J. Humaidi1. Aircraft Engines Remaining Useful Life Prediction Based on A Hybrid Model of Autoencoder and Deep Belief Network.
8. Lijun Liu1,2 · Lan Wang1, Zhen Yu1.Remaining Useful Life Estimation of Aircraft Engines Based on Deep Convolution Neural Network and LightGBM Combination Model.
9. Vimala Mathew, Tom Toby, Vikram Singh, B Maheswara Rao, M Goutham Kumar. Prediction of Remaining Useful Lifetime (RUL) of Turbofan Engine using Machine Learning.
10. Hussein A. Taha, Ahmed H. Sakr, Soumaya Yacout. Aircraft Engine Remaining Useful Life Prediction Framework for Industry 4.0.
11. Y. Peng, H. Wang, J. Wang, D. Liu, X. Peng. A modified echo state network based remaining useful life estimation approach. *IEEE International Conference on Prognostics and Health Management*. IEEE, 2012,pp. 1-7.
12. C. Louen, S. X. Ding, C. Kandler. A new framework for remaining useful life estimation using Support Vector Machine classifier. *Conference on Control and Fault-Tolerant Systems*, 2013, pp. 228-233.
13. G. S. Babu, P. Zhao, X. L. Li. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life.*International Conference on Database Systems for Advanced Applications*. Springer, 2016, pp. 214-228.
14. S. K. Singh, S. Kumar, J. P. Dwivedi. A novel soft computing method for engine RUL prediction. *Multimedia Tools and Applications*, 2017, pp. 1-23.
15. S. Zheng, K. Ristovski, A. Farahat, C. Gupta. Long Short-Term Memory Network for Remaining Useful Life estimation. *IEEE International Conference on Prognostics and Health Management*. IEEE, 2017, pp. 88-95.
16. P. Lim, C. K. Goh, K. C. Tan. A time window neural network based framework for Remaining Useful Life estimation. *International Joint Conference on Neural Networks*. IEEE, 2016, pp. 1746-1753.