The Netherlands Press

Journal of Airline Operations and Aviation Management

Article

PREDICTION OF AIRCRAFT ENGINE FAILURE USING RECURRENT NEURAL NETWORKS

[1]Kusuma Kurma, [*2]Sai Shankar

[1]Department of Information Technology Project Management,
St. Francis College, UNITED STATES,

Orchid ID: https://orcid.org/0000-0003-1458-883X

Email: kkurma@sfu.edu

[2]Department of Computer Science,
University of Dayton, UNITED STATES,

Orchid ID: https://orcid.org/0000-0002-0006-6400

Email: kurmal1@udayton.edu

Abstract.

The primary difficulty in aviation is evaluating the life of aircraft engines (AFs) in order to ensure that people on board and precious goods are transported safely from one nation to another. other nations at the port of arrival To foresee and anticipate all of these possibilities, we suggest combining AI and Deep Learning with a short-term memory (LSTM) neural network to forecast when the aircraft's engine will need to be fixed or replaced. The collection is needed of past aircraft history data with 21 sensor readings for each aircraft to make these predictions. Aircraft have three alternative settings = s1,s2,s3 so that data may be manipulated and the relationship / trend in the data can be discovered, allowing our system to forecast the remaining usable life (RUL) of an aircraft engine.

Keywords: Remaining useful life, Sensors, Long short term memory, Recurrent Neural networks.

Journal of Airline Operations and Aviation Management Volume 1 Issue 1
Received Date: 08 May 2022
Accepted Date: 15 June 2022
Published Date: 25 July 2022

1. Introduction

Estimating the remaining usable life is one of the purposes of predictive maintenance. The study of estimating when something will break based on its current status is known as remaining usable life (RUL). Figure 1 depicts the degradation of a machine over time as an example. The blue dot depicts the current state of a machine, as well as the remaining usable life. It's up to the red dot, and the red dot indicates the machine's failure state[1].

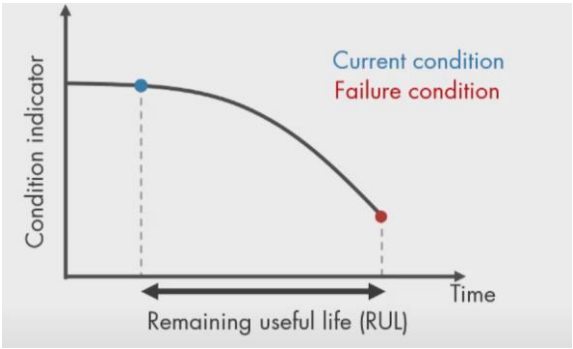

Figure 1: Machine deterioration profile [1]

The time interval between the current location and the failure point might be expressed in miles, cycles, days, or any other amount, depending on the system. Survival, degradation, and similarity models are three typical methods for estimating remaining usable life (RUL). The inputs determine which model to employ, such as whether the data is from a healthy condition to a failure state, if the data is solely at the moment of failure, or whether the data contains a safety threshold that should not be surpassed. When comparing data from a healthy state to a failed state, a similarity model is used. If the data solely contains failure portions, a survival model is used. If the data contains a safety threshold that should not be surpassed, a degradation model is used. The real question is how many more flights the engine can handle before its components need to be replaced. The survival mode is employed in this model if the data is solely failure conditions. It utilises the probability distribution to estimate the remaining usable life (RUL). When safety data is provided and the safety threshold is established, the degradation model is applied to the condition indicator, which utilises historical data to forecast how the condition indicator will change in the future[2]. This is a method of calculating the number of cycles till the condition indicator reaches the threshold, allowing us to estimate the remaining usable life. Another method for estimating RUL is to utilise a similarity model, which requires complete data from the health state to the failure state and is applied to failure histories from comparable machines. This data includes the whole fleet's history[3].

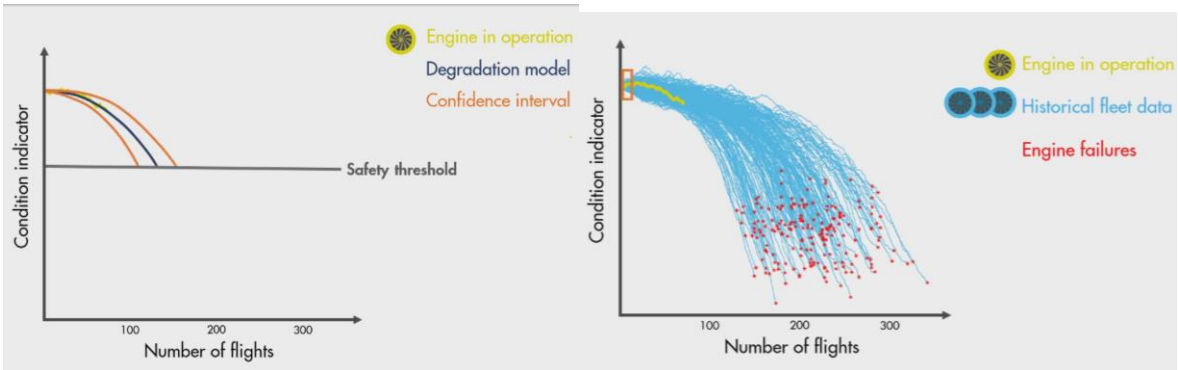

Figure 1: Degradation and similarity models[2]

2. PREDICTION MODEL OF AIRCRAFT ENGINE FAILURE:

For the prediction of aircraft life engine python is used and some of the libraries are required to install before starting the program. The libraries to be installed are

- Pandas
- Numpy
- seaborn
- matplotlib
- tensorflow api
- scikit learn

These are the libraries that should be imported in the workspace and importing the functions from the libraries.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from sklearn import preprocessing
from sklearn.metrics import recall_score
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras.layers import Dense, Dropout, LSTM, Activation
%matplotlib inline
```

2.1. Recurrent Neural network:

The model utilised in this case was recurrent neural networks, which consists of consecutive layers with two LSTM layers and a dense output layer with a single neuron, as shown in the diagram below[4].

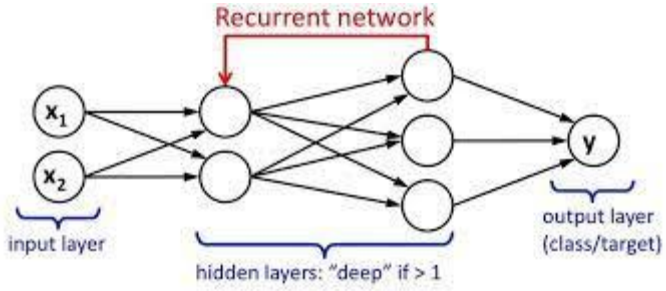

Figure 3: Neural network[5]

3. IMPLEMENTATION:

The primary goal of aviation engine failure prediction is to determine whether or not a certain aircraft engine will fail within the next 30 cycles. As a result, if an aircraft part fails, it will be replaced.

3.1. TAKING THE DATASET:

The data set used to forecast aeroplane engine failure is as follows: Aircraft Engine Remaining Useful Life (RUL) data contains information about actual cycles remaining for each engine in test data, and Test and Training data contains

engine operating data aircraft to failure, and the test data contained aircraft engine operating data with no failure events recorded[6]. The training and testing data are depicted in Figure 3 below. The data is presented in the form of .csv. With the aid of the pandas library, the csv and columns of the dataset are delivered and presented.

	unit_id	cycles	set1	set2	set3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	se11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	NaN	NaN
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	NaN	NaN
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	NaN	NaN
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	NaN	NaN
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	NaN	NaN

	unit_id	cycles	set1	set2	set3	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	se11	s12	s13	s14	s15	s16	s17	s18	s19	s20	s21	s22	s23
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	21.61	553.90	2388.04	9050.17	1.3	47.20	521.72	2388.03	8125.55	8.4052	0.03	392	2388	100.0	38.86	23.3735	NaN	NaN
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	21.61	554.85	2388.01	9054.42	1.3	47.50	522.16	2388.06	8139.62	8.3803	0.03	393	2388	100.0	39.02	23.3916	NaN	NaN
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	21.61	554.11	2388.05	9056.96	1.3	47.50	521.97	2388.03	8130.10	8.4441	0.03	393	2388	100.0	39.08	23.4166	NaN	NaN
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	21.61	554.07	2388.03	9045.29	1.3	47.28	521.38	2388.05	8132.90	8.3917	0.03	391	2388	100.0	39.00	23.3737	NaN	NaN
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	21.61	554.16	2388.01	9044.55	1.3	47.31	522.15	2388.03	8129.54	8.4031	0.03	390	2388	100.0	38.99	23.4130	NaN	NaN

Figure 4: Train and testing dataset

3.2. PRE-PROCESSING THE DATA:

Features of the data are unit_id has values from 1 to hundred. These values correspond to 100 engines. Number of cycles over of engine is given by 'cycles'. The last cycle for a unit_id is where that engine failed. set1, set2, set3 are three different setting of engines. s1 to s23 are sensor data of 23 sensors. The data of Remaining useful time is combined with the training data with the descending order of the RUL data.

	RULmax	unit_id
0	112	1
1	98	2
2	69	3
3	82	4
4	91	5

Figure 5: Final Dataset

From the RUL max dataset subtract number of cycles per each record to get RUL. There is no data present in s22 and s23 column so the data column should be dropped for the cleaning of data. This should be repeated for the testing data because testing data also contains null in s22 and s23 columns. By observing or by query the minimum and maximum value of the data.

1. test set engine of id number 1 took minimum number of cycles 31 cycles to fail
2. test set engine of id number 49 took maximum number of cycles 303 cycles to fail
3. test set engine of id number 69 took maximum number of cycles 362 cycles to fail
4. test set engine of id number 39 took minimum number of cycles 128 cycles to fail

3.2.1. Scaling the data:

For scaling the data to passing into the recurrent neural network the values in between 0 to 1 gives more accuracy than the normal ones so the values in the data should convert into 0 to 1 using the function called minmaxscaler. This is the function in the scikit learn library which is use for pre-processing and the robust functions of the problem.

```
[ ]  # normalise data to get a range of 0 to 1
     from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     df.iloc[:,1:-1]=scaler.fit_transform(df.iloc[:,1:-1])
     dfTest.iloc[:,1:-1]=scaler.fit_transform(dfTest.iloc[:,1:-1])
```

3.3. Build a Recurrent neural network model:

Using the sequential model from the tensorflow models as a starting point. The LSTM layer is inserted as the first layer of the sequence, and the activity regularizer is set to L2 with a count of 0.01. The LSTM layer is applied once again without the sequences being returned. Dense layer with sigmoid activation function is the last layer. The model's output is either 0 or 1, with 0 indicating that the engine will fail before the next 30 cycles and 1 indicating that the engine will not fail in the next 30 cycles.

```
#Modeling
from tensorflow.keras import regularizers
batch70=70
opt = keras.optimizers.Adam(learning_rate=0.004)
model = Sequential()
model.add(LSTM(input_shape=(batch70, trainLSTM.shape[2]),units=100,return_sequences=True,activity_regularizer=tf.keras.regularizers.l2(0.01)))
model.add(LSTM(units=50,return_sequences=False,activity_regularizer=tf.keras.regularizers.l2(0.01)))
model.add(Dense(units=trainLabel.shape[1], activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=opt, metrics=[tf.keras.metrics.Recall()])
print(model.summary())
```

The loss function is Binary cross entropy, and the optimizer is Adam, with a learning rate of 0.004 for constructing the model. Recall is one of the model's metrics. The summary of the model is depicted in the diagram below.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 70, 100)           47600
_________________________________________________________________
lstm_8 (LSTM)                (None, 50)                30200
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 51
=================================================================
Total params: 77,851
Trainable params: 77,851
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 6: Model Summary

4. RESULTS AND DISCUSSIONS:

In the train and test datasets, there are 21 sensors. The trainset's distribution is rather bell-shaped, with 12 engines failing after 200 cycles, which is the maximum in the distribution. After 300 cycles, just a few engines failed.

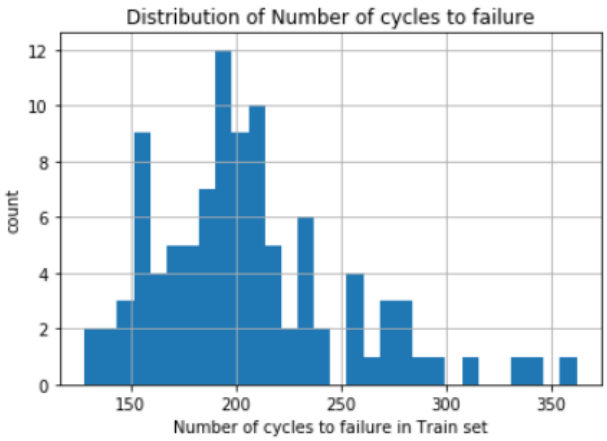

Figure 7: Distribution of number of cycles to failure in trainset

In the test set, however, 10 engines failed after 70 cycles. Which value in the distribution is the highest? And just a handful engines failed at 300 cycles.

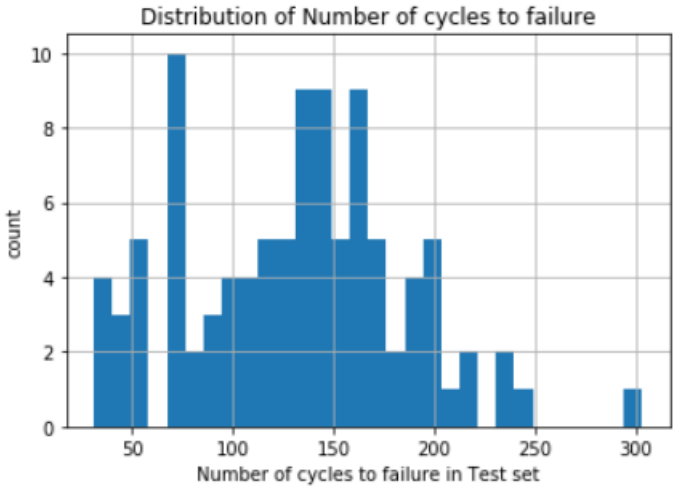

Figure 8: Distribution of number of cycles to failure in test set

The sensor data of 22 sensors of the engine ID 1 is shown in the graph below for viewing of the sensor data gathered for the engine. As a result, if the sensor data shows no substantial change in the trend between healthy and failure states, they will not contribute to the selection of important features for training a model similarity. As a result, data reduction is conducted in the pre-processing stage by choosing just the most trendable sensors and deleting the two columns of sensors that do not have any data by combining sensors to calculate condition indicators. The graphs below exhibit the sensor output behaviour for the train data with engine id 1 and 99, as seen in figures 8 and 9.

Figure 2: Sensor graph of engine ID 1

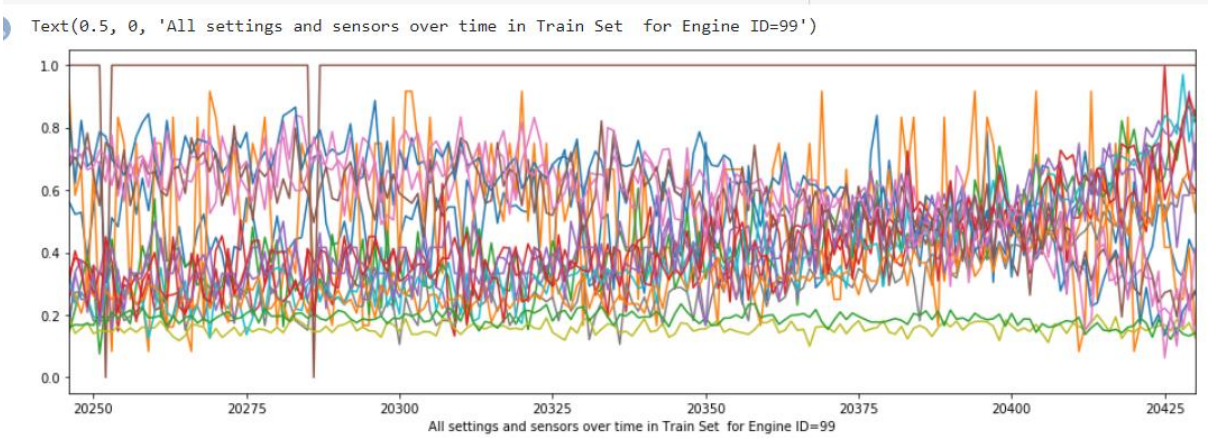

Figure 3: Sensor graph for engine ID 99

For the test set of engines for Engine Id of 100 and with RUL 20 the graph is plotted. For all the time the sensor output changes and hence the engine is considered for the testing purpose.

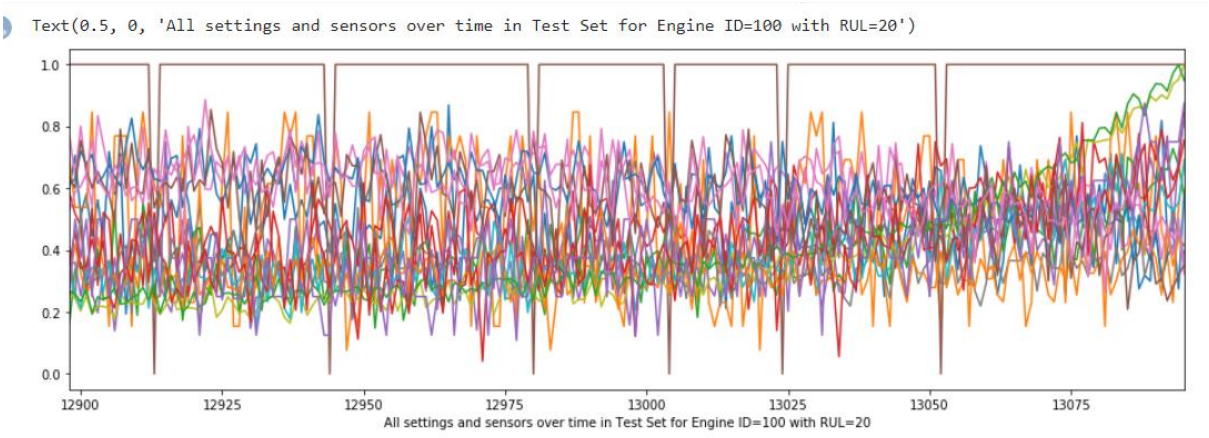

Figure 4: Sensor graph for engine ID 100

```
resultTrain=model.fit(trainLSTM, trainLabel, epochs=200, batch_size=256,
                      validation_split=0.25, verbose=1,
                      callbacks = [keras.callbacks.EarlyStopping(monitor='val_loss',
                                                                 min_delta=0, patience=20, verbose=0, mode='auto')])
```

To train the recurrent neural network model, which consists of two LSTM layers and a Dense layer. The validation and training losses are depicted in the figure, and the loss is strictly diminishing as indicated in figure 12, indicating that the model works effectively for aircraft engine failure.

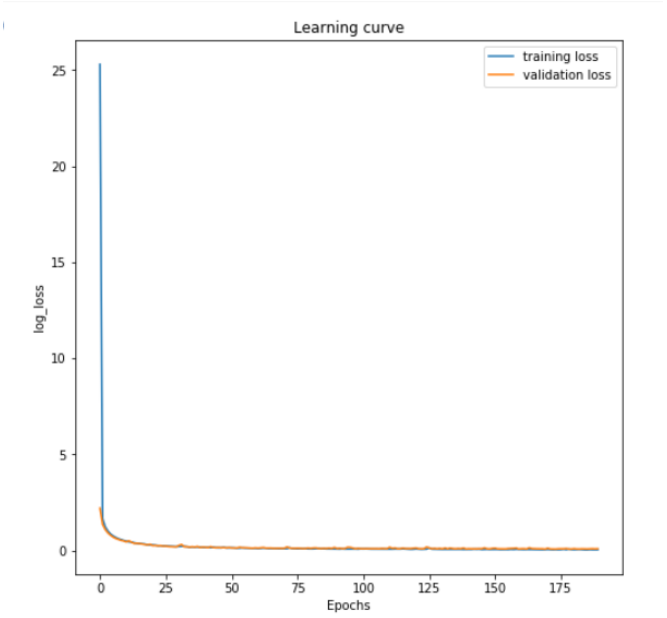

Figure 12: Training and validation graph

The test case is run through the model for assessment, and the result produced by testing the model is a recall of 90.8 percent, since the metrics indicated above is recall, thus the computed number is recall.

```
[ ]  #Evaluate
     resultTest = model.evaluate(testLSTM, testLabel)

     6473/6473 [==============================] - 8s 1ms/step

[ ]  print('Recall =',resultTest[1])

     Recall = 0.9082403182983398
```

5. CONCLUSION:

The RUL predicted by the LSTM neural network model for the subset exceeds the competition and achieves a higher retrieval value, demonstrating that our suggested LSTM neural network-based technique beats the competition and achieves a higher retrieval value. This data set closely resembles the real value. The test results show that the suggested approach can accurately estimate the RUL of aircraft engines. Furthermore, we may extrapolate that the LSTM neural network-based prediction technique outperforms the standard statistical probability regression method when dealing with huge amounts of data. It's also been argued that predicting the RUL of a unit with a brief history leads to a lot of uncertainty and bad projections. As a result, updating RUL projections is critical for effective planning.

References

[1] Vimala Mathew, Tom Toby, Vikram Singh, B Maheswara Rao, M Goutham Kumar, "Prediction of Remaining Useful Lifetime(RUL) of Turbofan Engine using Machine Learning" 2017 IEEE International Conference on Circuits and Systems(ICCS 2017). IEEE 2017.

[2] Dong Dong, Xiao-Yang Li, Fu-Qiang Sun "Life Prediction of Jet Engines Based on LSTM-Recurrent Neural Networks" School of Reliability and Systems Engineering, Science and Technology on Reliability and Environment Engneering. IEEE 2017.

[3] Mei Yuan, Yuting Wu and Li Lin "Fault diagnosis and Remaining useful life estimation of aero engine using LSTM neural network" 2016 IEEE/CsAA International Conference on Aircraft Utility System(AUS).

[4] Olgun Aydin, Screen Guldamlasioglu "Using LSTM Networks to Predict Engine Condition on Large Scale Data Processing Framework" 2017 4th International Conference On Electrical and Electronics Engineering.

[5] "Recurrent Neural Network Definition | DeepAI," DeepAI, May 17, 2019. https://deepai.org/machine-learning-glossary-and-terms/recurrent-neural-network (accessed Jun. 18, 2022).

[6] Zhi Lv, Jian wang ,Guigang Zhang, Huang Jiyang "Prognostic Health Management of Condition-Based Maintenance for Aircraft Engine Systems.

[7] A. Saxena and K. Goebel (2008). "Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository (https://ti.arc.nasa.gov/tech/dash/groups/p coe/prognostic-data-repository/#turbofan), NASA Ames