# RUL prediction using a R-CNN based approach

Rajan Pipaliya, Viral Pandey, Khilan Ravani and Manjunath V. Joshi

## Abstract

Prognostics refers to assessing and predicting the working condition of a machine components, based on its current and previous state, with the main motto being the accurate prognosis of Remaining Useful Life(RUL) of a machine or its part. It is extremely important in industrial domains where cost, reliability and safety are of utmost significance. With the development of various deep learning techniques, which redefine representation learning from raw data. Convolutional Neural Networks (CNN) tend to outperform other models while extracting position-invariant or salient features from a raw data, while Recurrent Neural Networks have prima facie advantages for modelling units in a sequence. Taking this into account, the technique proposed (RCNN) in this work focuses on integrating the CNN's with Long Short-Term Memory Networks (LSTM, a special type of RNN) to accurately predict the Remaining Useful Life (RUL) of a given degrading system. The proposed model is first trained on a multivariate time series data set (as provided in IEEE PHM 2008 challenge) to learn the hidden features and the engine degradation pattern, then tested on the actual data. The results obtained on these data confirm the potential of the proposed model in accurately predicting the RUL of a system.

***Keywords : RUL Prediction; Similarity measure; Data driven approach; RCCN based approach***

## Introduction

Breakdown of the machine part has a huge impact on the industrial systems and can lead to disastrous situations if routine maintenance of the equipment is carried out. Day by day, the machinery employed in fabrication is becoming more and more complicated. In these systems, reliability is highly essential because a single fault can become a cause for greater mishap. In order to avoid such situation, maintenance methods nowadays have transitioned from conventional methods like “fix when system breaks” to “Condition-Based Maintenance (CBM)”. CBM is a relatively newer maintenance philosophy established upon the concept of Prognostics and Health Management (PHM). It involves real-time evaluation of equipment sensor data to determine the system condition or health. This helps in scheduling maintenance activities on the basis of necessity as identified by the condition-based system and results in increased safety and reliability bringing down the maintenance costs.

Prognostics is defined as the estimation of time to failure and risk for one or more existing future failure modes. Prognostics based approaches can be categorized into three categories : Model based, Data-driven and Hybrid approach. Model-based approaches are used to build mathematical models connected to the physical processes. On the other hand, data-driven approaches are used to derive different data from the monitoring data rather than building the models based on system mechanics. They are learning based techniques built upon the historical records and provide prediction output directly in terms of health condition of the systems. Hybrid approaches correspond to combination of model based and data driven approaches. A common method in these approaches is to validate the physical model offline, then use data-driven techniques for prediction and also update the parameters to increase level of the accuracy in prediction.

Data-driven approaches offer different strategies to perform prognostics. Artificial Neural Networks(ANN) is the most commonly used data-driven approach. However, neural networks lack the ability to demonstrate and transcribe their results. Statistical approaches are also popular in the field of prognostics. However, they are usually based on assumptions that are not suitable for real world industrial applications, where reliability is of greater importance. In this work, a data-driven approach employing deep learning methods are developed. Deep learning methods outperform other methods if the data size is large. Deep learning methods have a ability to perform automatic feature extraction from raw data. Our proposed approach is based on deep learning, it gives an accurate prediction results without any prior knowledge about the component's characteristics or the monitored system. Proposed approach is a combination of CNN and RNN(RCNN). It outperforms the existing RNN based approach. The proposed approach can be used for any similar applications. Our approach makes the following assumptions for the model :

- The data is recorded from a random point of time when the system is healthy and ends when it stops working.

- We assume that testing systems exhibits the same degradation behaviour.

The rest of the paper is organized as follows. Section 3 presents related work, while in Section 4, we present the proposed model. Section 5 explains the architecture of the model and Section 6 describes the performed experiment/analysis. Section 7 demonstrates the performance of our method and baselines on real data-sets. Finally, the paper is concluded in Section 8. Refer to Section 9 for more experiment results and discussion.

## 3 Related Work

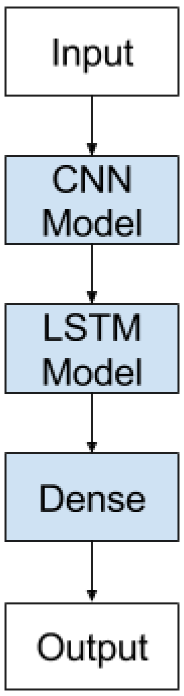
PHM is one of the important research topics these days and is receiving significant attention from various commercial enterprises such as transportation, military, aerospace etc. The methods which defer from using require mechanics-based models are trending and those using historical data or data-driven methods are increasingly being applied to encounter different challenges in this domain. As per techniques for computing the remaining useful life depends on the type and amount of data available for the particular modules under inspection. Various research articles have been recently published in this domain, with the goal of providing a thorough review of the available prognostics techniques and the related areas.

Different training models including SVMs, RNNs, CNNs have been tested for RUL prediction. Recurrent Neural Network (RNN) [7] can model time sequence data. Some work[8] applied RNN for RUL estimation. However, RNN is known to have long-term time dependency problems [9], where the gradients propagated over many stages tend to either vanish or explode.

Recently, Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), has been shown to be efficient in applications such as computer vision, image, video, speech[10], [11], [12]. A CNN based RUL estimation approach is proposed in [13], where the sensor data is split into sliding windows and then CNN is applied on each sliding window. In this case, CNN also considers each sliding window independently.
In this work, we explore a data-driven, prognostics approach based on the concept of blending CNN with LSTM . In our proposed model, LSTM and CNN are made to learn the trend evolution based on local raw data of time series and then our model combines the features captured by LSTM and CNN to predict the trend.

## 4 Proposed Approach :

The dataset used for training purpose contains data of identical engines. This data is natural time series data of different sensors and these are recorded from a time when the system is healthy to the point when any fault occurs. Different sensors including acceleration, vibration etc. are used for monitoring the system.
The proposed approach, as described in the given below figure, predicts RUL of test component well in advance of the actual 'failure state'. The idea is to compare the current degradation of test component with the data of the training components and thereafter, utilize the closest one to estimate the RUL of the respective test component.



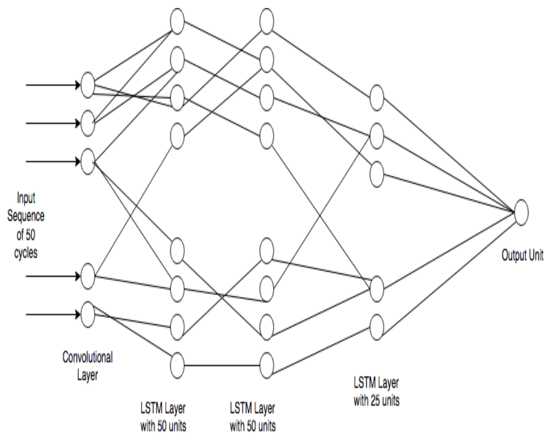
In pre-processing, the raw data (input feature vectors) is normalized and then fed as input to the CNN for feature extraction purposes in batches of size m x n where m is the the number of rows/life cycles of engine and n is the the number of columns/sensor data corresponding to different operational settings.

Using CNN, local patterns are identified more efficiently because the main assumption of the CNN model is that the local patterns are relevant everywhere. Hence, 1D CNN adopts multiple local filters over whole sequential input. Each feature map corresponding to each local filter can be generated by sliding the filter over sequence.Then pooling layer is applied to extract most vital and fixed length features from each feature map. Then, the output from CNN is fed to the LSTM layer.

CNN is able to encode more critical information compared to the raw input, considering that the single time step information may not be discriminative enough. In addition, CNN is able to compress the length of the sequence, which increases the capability of the subsequent LSTM to capture temporal information.

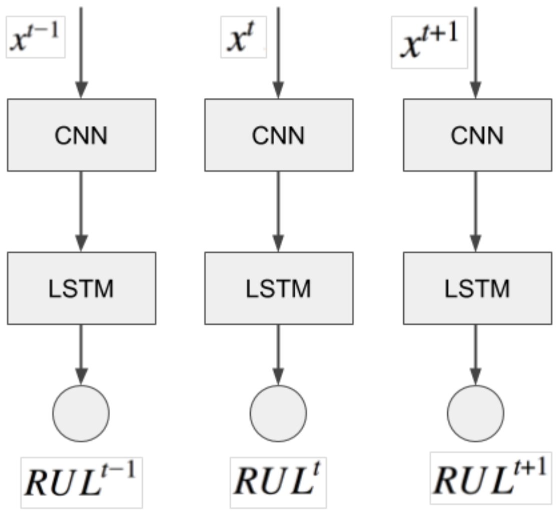
LSTM prevents the problem of backpropagation from vanishing or exploding. LSTM avoids long-term time dependency problems by controlling information flow using input, forget and output gates.[11] Remembering information for long periods of time is practically their default behavior. Due to inherent sequential nature of sensor data, LSTM are well-suited for RUL estimation using sensor data. LSTM processes input sequences in forward and backward directions and is able to summarize temporal information from past and future contexts, so now both past and future dependency information are used to capture the temporal information.

Dense layer is used to reduce the dimension of the output of LSTM layer, so that we get single vector as a output of the model.



## Architecture of RCNN model

Here, we assume that there are p sensors of the same type on each machine. Then data collected from each sensor can be represented in a matrix from $X_n = [x^1, x^2, ...., x^{T_n}]$, where $T_n$ is time to failure and $x^t = [x^t{}_1, x^t{}_2, ...., x^t{}_p]$ (where t = 1,2,.... $T_n$) is p-dimensional vector of sensor measurements at time t. Here for each $X_i$, the corresponding tool wear condition is measured as $Y_i$. The task is to predict final $Y_i$ based on sequential data $X_i$.



### Convolutional Layer

CNN layer is first layer in the model and it takes specific size batch of $X_n$ as an input. A classical CNN has three cascaded layers. Starting with convolution, pooling, followed by activation. CNN layer slides the filters over the window of the input sequence to generate feature maps. It is assumed that there are k filters with the window size m. Then activation function and pooling layer is applied. This three layers are described as follows :
The Convolution layer is defined as multiplication operation between input vector $X_{i:i+m-1}$ and filter vector $u \in R^d$ where d is number of sensors .

$$X_{i:i+m-1} = x_i * x_{i+1} * x_{i+2} * .... * x_{i+m-1}$$

Here $X_{i:i+m-1}$ represents a window size of m starting from i-time step. Here bias term b is also added so the final convolution operation is :

$$c_i = u^T x_{i:i+m-1} + b$$

Then non-linear activation function Relu is applied on output of convolution operation:

$$c_i = g(c_i)$$

In CNN convolution operation is applied on the sequence by sliding the filtering window from beginning to the end of the sequence. So vector of feature map looks like below sequence:

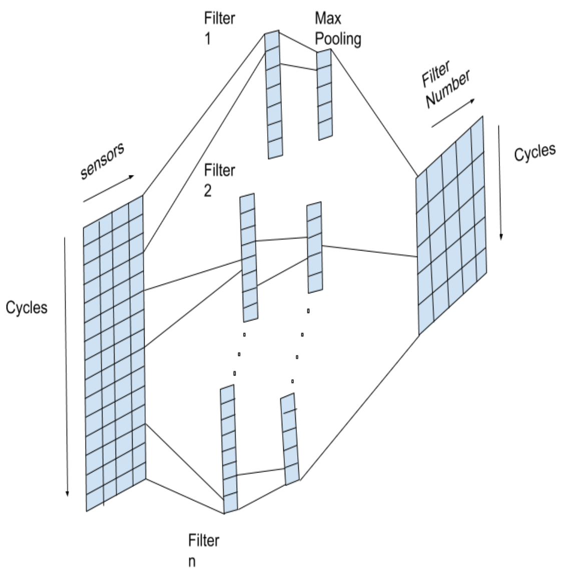
$$c_j = [c_1, c_2, c_3, .... * c_{l-m+1}]$$

Here, $l$ is length of sequence and $j$ shows the jth filter.

Pooling operation is able to reduce the length of the feature map, that can minimize the number of model parameters. $s$ is denoted as hyper parameter of pooling operation. So it takes maximum value from the $s$ consecutive values from the feature map $c_j$.
Output of the pooling operation is compressed feature vector and it is obtained as:

$$h = [h_1, h_2, h_3, .... * h_{(l-m)/s+1}]$$

In the CNN layer size of input sequence is $n * l * d$ where n is number of data samples, l is length of sequence and d is total number of sensors and the size of the output is $n * k * ((l-m)/s+1)$. Here it can see that after CNN layer, the length input sequence is reduced from $l$ to $(l-m)/s+1$ Therefor, CNN do the role of feature extractor to feed better sequential data into LSTM layer compare to the raw sequential data.



## Recurrent layer

Here, a Long Short-Term Memory (LSTM) a special type of RNN architecture is implemented.
LSTM cell state which is its main function, has different states which determines as to which data from the foregoing cell state to be taken in consideration or be neglected (forget state), what to update from the incoming data (input state) and decide as to whether the cell state would affect other neurons or not (output state).
In LTSM layer, it performs multiple internal equations as described below. In these,

$$i_t = (W_i x_t + W_i h_{t-1} + b_i)$$
$$o_t = (W_O x_t + W_O h_{t-1} + b_O)$$
$$f_t = tanh(W_f x_t + W_f h_{t-1} + b_f)$$
$$a_t = tanh(W_a x_t + W_a h_{t-1} + b_a)$$
$$C_t = i_t * a_t + C_t * f_{t-1}$$

$h_t = o_t * tanh(C_t)$

$x_t$ = recurrent layer input, $i_t$ = , $o_t$ =, $f_t$ = , $C_t$ = , $h_t$ = final output, $W$ = weight matrices $h_t$ .

In this model, there is multiple LSTM layer are used. One issue with LSTM modelling is that it overfits training data and reduces own prediction accuracy. To reduce overfitting, we introduce dropout in model which has a probability of 0.2.

### Output layer

RCNN model's end step is a layer that is completely linked. It has linear activation function.

In this approach, the model has total five layers. First layer is convolutional layer and it has same properties those are described above. 1D CNN layer has 50 filters, filter size is 3. It uses Relu as activation. Second, third, fourth are LSTM layers and final fifth layer is dense layer. Second and third LSTM layers are bidirectional LSTM and have 100 units and their dropout is 0.2. Fourth layer has total 50 unit and it's dropout is 0.2 . LSTM layers use linear activation function.

We also add Adam as optimizer and to calculate loss we use standard mean square error function.

## Experiment

To test the model, we used PHM08 and CMAPSS datasets. And we compared model(CRNN) with RNN model.

### Data Description

In PHM08 dataset[23], Each line of dataset is a snapshot of cycle. There are total 26 columns. Columns represent engine number, cycle number, three different operational settings(OP) and then 21 different sensor value in order .At each cycle sensors read reading and their values are stored between columns 3 to 26 :

1) Engine number
2) Cycle number
3) OS 1
4) OS2
5) Os3
6) S1
7) S2

.

.

26) S21

As long as the prognostics algorithms such as linear regression, CNN, RNN and RCNN will be used, the physical meaning of each attribute is trivial. The problem here is to understand how the data could be modeled with a suitable data driven algorithm.

PHM08 data set contains total three data files. **218** identical engines are used to create the training dataset. They are run until they stop working. These engines had different lengths of cycles and varied from ***127*** to ***257.***

Similarly, testing dataset has 100 identical engines. Task on hand is to predict the remaining life of an engine before it stops working.
In truth dataset, actual life of test engines are given.

### Sensors Selection

From data analysis, sensors that contributed to the degradation of the engine by showing some trend are considered in final input dataset. Based on this idea, there are total 13 sensors and 2 operanal settings are used in input dataset of model. [2]

## Normalization

To get uniform range values to each of the inputs, normalization is done . Below function is used to normalize the input data:

$$Norm(x^f) = (x^f - \mu^f)/\sigma^f, \forall f$$

Where $f$ represents each of the original 21 sensor, $\mu^f$ is mean and $\sigma^f$ is standard deviation of sensor $f$.

Now datasets are ready for training process. We trained datasets on the model that is described above.

## Evaluation function

Evaluation(score) function is applied for performance measure of the model. This evaluation function is used to compute model in PHM08 competition[8]. It computes to penalize late predictions seriously rather than prior predictions. As faults increases, penalty would grow exponentially. The parameters *x1* and *x2* in the function that calculates final score controls the preferences.

$$s = \Sigma^n_{i=1} e^{(d/x_1)} - 1, for\ d \geq 0$$
$$s = \Sigma^n_{i=1} e^{-(d/x_2)} - 1, for\ d < 0$$

Here, *s* = engine score, *n* = Engines, d = Estimated RUL – Actual RUL, *x*1 = 13, and *x*2 = 10.

## Results

This model gave a late prediction for 30 units, early prediction for 48 units, and exact output for 16 units out of 93 test units. Overall score for model is 292.690. Following image show actual out vs predicted output value for testing data set.

For training data set,

**MAE: 10.8782712581**
**R^2: 0.876296297945**

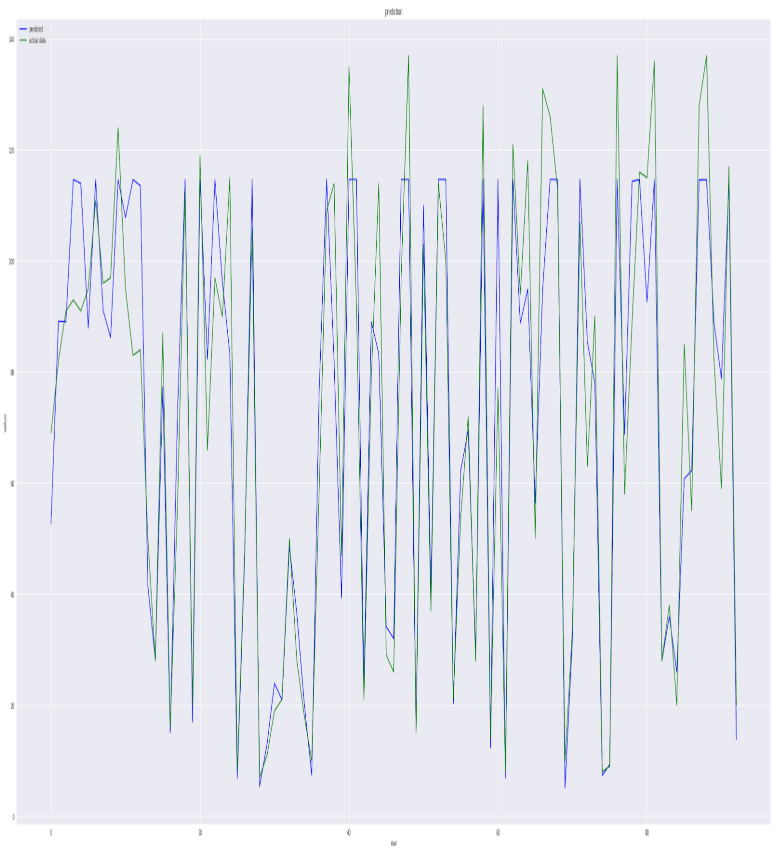
After prediction of remaining life cycle of testing dataset. For testing data set,
**MAE: 12.8186536142**
**R^2: 0.806095148927**

We have also compared proposed model with other models and their score are mentioned in below table.

| Algorithm | Score |
|---|---|
| ANN | 3212 |
| SVR | 15886 |
| CNN | 2056 |
| LSTM | 1863 |
| RCCN | 1285 |

## Conclusion

We have proposed an approach in deep learning that combines CNN and RNN for RUL prediction. RCNN model is able a make local relation of sequence of sensors value.



## References:

[7] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[8] F. O. Heimes, “Recurrent neural networks for remaining useful life estimation,” in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*. IEEE, 2008, pp. 1–6.

[9] Y.Bengio,P.Simard,andP.Frasconi,“Learning long-termdependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[10] G.E.Hinton,S.Osindero,andY.-W.Teh,“Afastl earningalgorithmfor deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] S. Zheng, A. Vishnu, and C. Ding, “Accelerating deep learning with shrinkage and recall,” in *Parallel and Distributed Systems (ICPADS), 2016 IEEE 22nd International Conference on*. IEEE, 2016, pp. 963–
computing resource for the model in Figure 1. If there is high 970.

[13] G. S. Babu, P. Zhao, and X.-L. Li, “Deep convolutional neural network based regression approach for estimation of remaining useful life,” in*International Conference on Database Systems for Advanced Applica- tions*. Springer, 2016, pp. 214–228.

[20] Hochreiter S, Schmidhuber J, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[23] A. Saxena and K. Goebel, “Phm08 challenge data set,” *NASA Ames Prognostics Data Repository (http://ti. arc. Nasa. gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA*, 2008.