Deep Recurrent Architecture with Attention for Remaining Useful Life Estimation

Ankit Das*, Shaista Hussain*, Feng Yang†, Mohd Salahuddin Habibullah‡, Arun Kumar§
*,†,‡ Institute Of High Performance Computing, Singapore
§ Department of Computer Science and Engineering, NITR, India
Email: *,†,‡ {ankit_kumar_das, hussains, yangf, mohdsh}@ihpc.a-star.edu.sg § {kumararun@nitrkl.ac.in}

***Abstract*—In many industries, there is a growing awareness to ensure the reliability and availability of manufacturing systems. Monitoring the health of machines enables the users to schedule repair and maintenance of the system ensuring less downtime, thereby enhancing its lifetime. With the advent of sensor technology, machine learning based algorithms show promise in estimating the Remaining Useful Life (RUL) of machine components. This paper presents recurrent architecture with attention for estimating the RUL of turbofan engines. First, we present a Deep Long Short Term Memory (DLSTM) network with dropout at multiple layers for RUL prediction. Next, we enhance the DLSTM model to handle the sequence in both forward and reverse direction using a Bidirectional Deep Long Short Term Memory (BiDLSTM). Finally, we present an Attention based Deep LSTM (Attn-DLSTM) which takes into account all the timesteps in estimation of the RUL. The inclusion of attention mechanism helps improve the accuracy as well as interpretability of the deep LSTM network. All the experiments are carried out using the publicly available NASA turbofan dataset. Results show the efficacy of deep networks compared to traditional machine learning algorithms.**

Keywords - Remaining Useful Life, Long Short Term Memory, Bidirectional Recurrent Neural Networks, Attention Mechanism.

I. Introduction

In advanced manufacturing industries, estimating the Remaining Useful Life (RUL) of machine or its components is one of the key problems. Estimating the RUL enables to plan the repair and maintenance of the machine, thereby reducing its downtime and saving cost of operations. In recent years this has been made possible due to the advent of better and cost effective sensor technology which enables users to collect failure data of machines by installing sensors. This allows researches to perform condition monitoring and RUL estimation.

The current literature in RUL prediction can be classified into two broad categories namely modeling based approaches [1] and data-driven approaches [2]. In modeling based approach, the focus is to develop a mathematical model of the system in order to predict the RUL. However, it is often difficult to completely characterize the system mathematically. In data-driven approach, the focus is to learn the mapping between the condition monitoring data and RUL of the machine or its components using different machine learning algorithms. The majority of works in the literature belong to the data-driven category due to low cost of sensors to record data and development of machine learning algorithms to process it.

Data-driven approaches in field of prognostics generally follow the traditional data analysis pipeline of feature extraction, feature selection followed by modeling and prediction. However, majority of these models depend on handcrafted features to represent the data. This not only limits the expressive power of models but also requires expert knowledge which may not be available always. The other class of data driven algorithms popularly known as deep learning algorithms takes raw data and extracts features directly without the need of expert knowledge [3].

In recent years many deep learning based algorithms have been proposed for different tasks in natural language processing [4], image processing [5] etc. In [6], a Convolutional Neural Network (CNN) based algorithm has been employed for RUL prediction. Here, the CNN algorithm is used to extract features by dividing the data into segments via a sliding window based approach. However, this algorithm does not take into account the sequence information into consideration. Recurrent Neural Network (RNN) [3] are a class of algorithms which represent the predicted variable as function of current data as well as past information. However, RNN's suffer from known problems such as vanishing and exploding gradients [7]. Long Short Term Memory (LSTM) networks [8] are a class of RNN's which have the capability of learning the long term dependency information in sequences of data. In recent literature, LSTM's have been employed for RUL prediction [9]. Another variant of LSTM, known as Bidirectional Long Short Term Memory (BiLSTM), considers the sequence of data in both forward and reverse order, in other words exploits both past and future information for RUL prediction [10]. However, both LSTM and BiLSTM put more emphasis on last time step while predicting the RUL due to which the influence of earlier time steps might diminish. Also, it is not clear which combination of time steps has a larger influence on the final prediction.

Attention mechanism has shown success in different domains like speech, NLP, healthcare [11], [12] etc. The attention mechanism has the ability to take into account all the time steps while making the prediction. This also enhances the interpretability along with accuracy of network. In this paper, we present deep recurrent architectures for RUL prediction.

978-1-7281-1895-6/19/$31.00 ©2019 IEEE

First, we present a two layer Deep LSTM (DLSTM) with dropout at both the layers to predict the RUL using the NASA turbofan dataset [13]. Next, we enhance the DLSTM network by propagating the input sequence in both forward and reverse manner by employing a Bidirectional DLSTM (BiDLSTM). Finally, we present an Attention based DLSTM (Attn-DLSTM) network which provides emphasis to all the time steps while computing the RUL. The experimental results show that Attn-DLSTM is able to achieve better accuracy in most of the cases. The performance of the presented deep recurrent architectures is also compared with traditional machine learning algorithms like Multilayer Perceptron (MLP) , Support Vector Regression (SVR), Relevance Vector Regression (RVR) and CNN in deep learning category. Results show that the deep recurrent architectures perform better than traditional machine learning algorithms and CNN in most of the cases.

The rest of the paper is organized as follows. In Section II, the different architectures employed in this study are explained in detail. Section III shows the results of various architectures and comparison with other algorithms. Section IV provides the conclusions from this study.

II. Methodology

In this section, we describe the details of different recurrent architectures presented in this paper. Recurrent Neural Networks (RNN) capture the sequential information by updating the hidden state for each input. The final output in recurrent architectures is a function of current input and hidden state from the previous cycle of input. In this work, we present three different architectures, where the building block of all these architectures is the Long Short Term Memory (LSTM). First, we describe the DLSTM followed by BiLSTM and Attn-LSTM.

A. Deep Long Short Term Memory

The LSTM based neural networks are a class of RNN's which learn the input data by recursively updating its state variable. However, RNN's suffer from the vanishing and exploding gradient problem [14]. An LSTM based network addresses these problems using a more sophisticated gating mechanism which allows the network to learn longer time dependencies. Fig. 1 shows an LSTM cell [8] at a particular time instance t. The input gate, output gate, forget gate which form the core of an LSTM cell are denoted by i_t, o_t and f_t respectively. The memory cell is denoted by c_t and the output of the LSTM cell is denoted by h_t. The information flow in the LSTM cell is characterized by following equations (1). The weight matrix and biases of input gate, output gate, forget gate and memory cell are denoted by W_i, W_o, W_f, W_c and b_i, b_o, b_f, b_c, respectively. All these trainable parameters are shared between all the time steps and are tuned while presenting the network with training samples. σ represents the sigmoid activation and $\circ$ denotes the element wise multiplication operator.

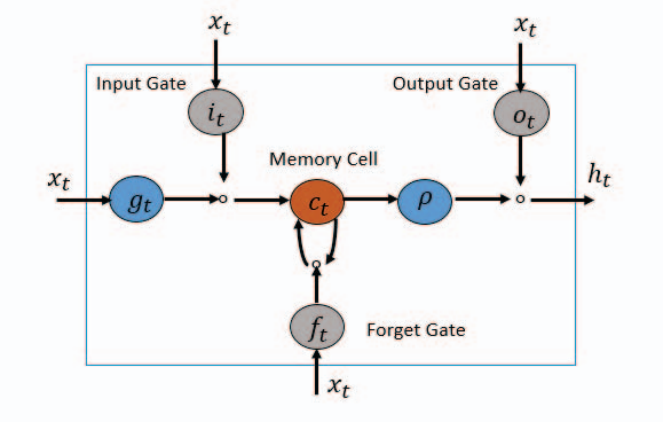

Fig. 1: LSTM Cell.

$$
\begin{aligned}
g_t &= tanh(W_c x_t + U c h_{t1} + b_c) \\
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ tanh(c_t)
\end{aligned} \tag{1}
$$

In this paper, we employ the DLSTM architecture illustrated in Fig. 2 for RUL estimation. Here, the input data sequence is represented by $\{x_1, x_2, ..., x_{t-1}, x_t\}$. We construct the architecture by stacking up LSTM cells in two layers. We employ a dropout layer after each LSTM layer to avoid overfitting the data. Finally, the output of the last time step is passed onto a fully connected layer (eq 2) to predict the output.

$$\hat{y} = W h_t^T \tag{2}$$

where W, h_t denotes the weight of fully connected layer and output of the last time step of DLSTM.

We employ the Mean Squared Error (MSE) loss for training the network given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\bar{y}_i - y_i)^2 \tag{3}$$

where n is the number of samples in the training dataset.

The training of the DLSTM is performed in a supervised fashion. First, the weights of the network are initialized randomly. In the forward pass the output of the network is obtained. The output is then compared with the actual output and MSE loss is computed. The gradients of the network are then computed with respect to the loss. This process is carried out for a certain number of epochs to allow the network to converge. In order to improve the computational efficiency of the training process of the DLSTM network, the training data is divided into batches.

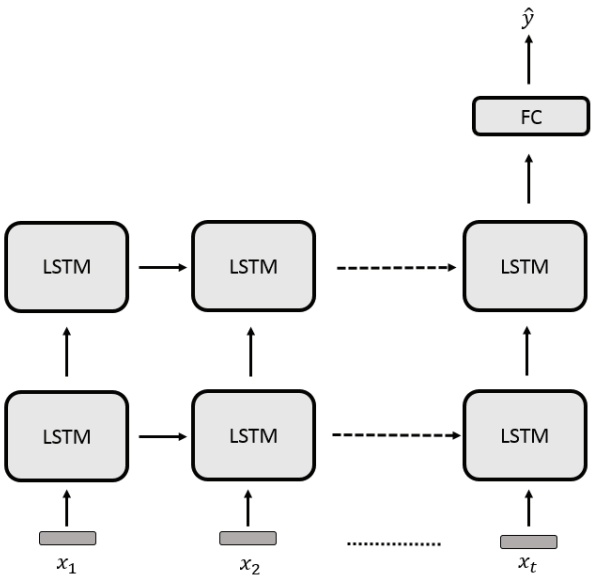

Fig. 2: Deep LSTM.

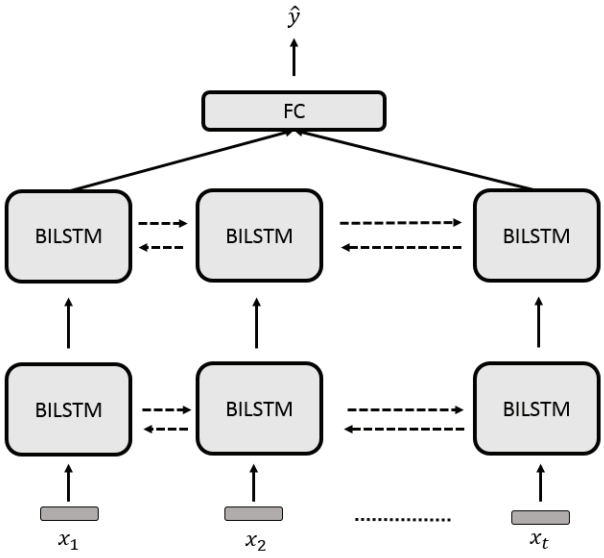

Fig. 3: Bidirectional Deep LSTM.

B. Bidirectional Deep Long Short Term Memory

The DLSTM network described above has the capability to capture the sequential information. However, in sequence learning tasks it might be beneficial to consider both the past as well as future information for prediction. Bidirectional Long Short Term Memory (BiLSTM) networks [10] are an extension of LSTM which have the capability to process the sequence in both forward as well as reverse directions. The basic idea of BiLSTM is that two different layers of network have been designed to learn and process the information in both forward and backward manner, which enables the network to make predictions by capturing both the past as well as future dependencies. The equations (4) and (5) describe the forward and reverse propagations. To be consistent with DLSTM, we stack two layers of BiLSTM blocks with dropout after each layer to build the BiDLSTM network. Here also, we employ the MSE loss defined in equation (3) for training. The architecture for BiDLSTM is shown in Fig. 3. The trade-off between the two architectures is that BiDLSTM employs double the number of parameters compared to that of DLSTM.

$$
\begin{aligned}
\overrightarrow{g}_t &= tanh(\overrightarrow{W}_c \overrightarrow{x}_t + \overrightarrow{U}_c \overrightarrow{h}_{t-1} + \overrightarrow{b_c}) \\
\overrightarrow{f}_t &= \sigma_g(\overrightarrow{W}_f \overrightarrow{x}_t + \overrightarrow{U}_f \overrightarrow{h}_{t-1} + \overrightarrow{b}_f) \\
\overrightarrow{i}_t &= \sigma_g(\overrightarrow{W}_i \overrightarrow{x}_t + \overrightarrow{U}_i \overrightarrow{h}_{t-1} + \overrightarrow{b}_i) \\
\overrightarrow{o}_t &= \sigma_g(\overrightarrow{W}_o \overrightarrow{x}_t + \overrightarrow{U}_o \overrightarrow{h}_{t-1} + \overrightarrow{b}_o) \\
\overrightarrow{c}_t &= \overrightarrow{f}_t \circ \overrightarrow{c}_{t-1} + \overrightarrow{i}_t \circ \overrightarrow{g}_t \\
\overrightarrow{h}_t &= \overrightarrow{o}_t \circ tanh(\overrightarrow{c}_t)
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
\overleftarrow{g}_t &= tanh(\overleftarrow{W}_c x_t + \overleftarrow{U}_c \overleftarrow{h}_{t-1} + \overleftarrow{b}_c) \\
\overleftarrow{f}_t &= \sigma_g(\overleftarrow{W}_f \overleftarrow{x}_t + \overleftarrow{U}_f \overleftarrow{h}_{t-1} + \overleftarrow{b}_f) \\
\overleftarrow{i}_t &= \sigma_g(\overleftarrow{W}_i \overleftarrow{x}_t + \overleftarrow{U}_i \overleftarrow{h}_{t-1} + \overleftarrow{b}_i) \\
\overleftarrow{o}_t &= \sigma_g(\overleftarrow{W}_o \overleftarrow{x}_t + \overleftarrow{U}_o \overleftarrow{h}_{t-1} + \overleftarrow{b}_o) \\
\overleftarrow{c}_t &= \overleftarrow{f}_t \circ \overleftarrow{c}_{t-1} + \overleftarrow{i}_t \circ \overleftarrow{g}_t \\
\overleftarrow{h}_t &= \overleftarrow{o}_t \circ tanh(\overleftarrow{c}_t)
\end{aligned}
\tag{5}
$$

C. Attention based Long Short Term Memory

From the above discussion, we know that LSTM and BiLSTM based networks aim to learn the input sequence in unidirectional and bidirectional manner respectively. However, it can be seen from Figs. 2 and 3 that these networks put emphasis on the output of the last time step in order to make the prediction. As a result of this the network might lose the information from the earlier time steps. The last hidden output might contain only partial information needed for prediction. In order to address this drawback, we employ attention mechanism to also give weightage to earlier time steps for prediction. The attention mechanism has been shown to work well in other tasks, as discussed in [12], [15].

In this work, we present an Attention based Deep LSTM network (Attn-DLSTM) shown in Fig. 4. It can be seen from the figure that the network takes into account the activations from all the time steps. The attention mechanism is employed after the second layer of LSTM cells. We pass the hidden outputs from all the time steps to a fully connected layer followed by a softmax layer to generate attention weights as follows:

$$\alpha_i = W_\alpha h_i + b_\alpha \tag{6}$$

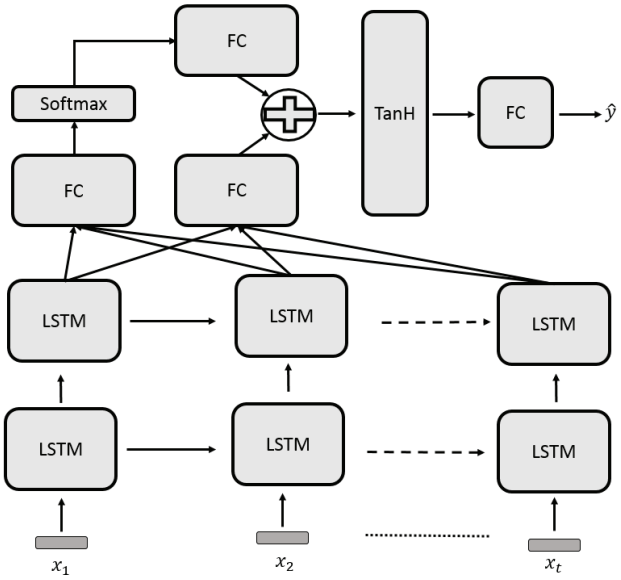

Fig. 4: Attention Based Deep LSTM Architecture.

$$\alpha_h = softmax[\alpha_1, \alpha_{t1}..., \alpha_t] \tag{7}$$

where, α_h denotes the weight for all the time steps of the network. Next, we compute the weighted sum of all the time steps given by:

$$p = H\alpha_h \tag{8}$$

where, H denotes the output from all the hidden states.

The weighted sum p along with output from the last time step is then passed through a fully connected layer followed by a $tanh$ activation function as follows:

$$\hat{h} = tanh(W_p p + W_h h_t) \tag{9}$$

Finally, the predicted output is computed using the following equation:

$$\hat{y} = W_l \hat{h} + b_l \tag{10}$$

III. Experimental Results

In this work, we applied the presented recurrent architectures DLSTM, BiDLSTM and Attn-DLSTM for predicting the RUL of turbofan engines. We present the results of varying network structure leading to the selection of network hyperparameters. We employ the square root of MSE (RMSE) to compare the performance of all the algorithms. To understand the interpretability of the model, we analyze the attention weights of the attention based DLSTM. The results obtained in this work using these three methods were also compared with the performance reported in [6].

To choose the best network architecture, we performed 10-fold cross validation (CV) on training data. We divided the training set into batches of 200 samples and then the validation MSE values averaged over all batches were computed for each CV fold by training the network for 50 epochs. We then used the trained architecture with the best validation result for predicting the RUL on the test data. We employed the RMSprop optimizer [3] to tune the parameters of the networks. The RMSprop optimizer is similar to the gradient descent algorithm with momentum, which helps the algorithm to converge faster. Moreover, the RMSprop includes another parameter to prevent the gradients from blowing up. Our experimental platform is a server with NVIDIA GeForce RTX 2080 Ti GPU with Ubuntu 18.04 operating system. The programming language used in this work is Python 3.7 with deep learning library PyTorch (1.1.0).

Dataset	FD001	FD002	FD003	FD004
Training trajectories	100	260	100	249
Test trajectories	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

TABLE I: C-MAPSS dataset from NASA data repository [13]

A. Data

The data employed in this work is Turbofan engine degradation run-to-failure datasets generated using C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) [13]. This dataset consists of four sub-datasets consisting of multiple multivariate time series data collected from 21 sensors and 3 operating condition measurements, where each time series corresponds to a specific turbofan engine data taken during a single operating time cycle. As shown in Table I, each sub-dataset with different number of operating conditions and fault conditions is further divided into training and test sets. Each row of time series in these datasets includes 26 columns consisting of engine ID, operational cycle number, three operational settings and 21 sensor measurements. Training datasets comprise data where fault grows with each operational cycle until system failure while test datasets contain data up to some time prior to the system failure. Therefore, the task for the C-MAPSS dataset involves estimating the remaining number of cycles before failure for the test data.

B. Preprocessing

The sensor data features and operational conditions data can have different scales which will lead to unequally weighted input data features being fed to the network. Hence, before training the model, a normalization step was performed to convert all features into a common scale. For this purpose, all sensor and operating condition features were normalized using z-score normalization, which was done by subtracting the population mean from each input sample and then dividing by the population standard deviation. To leverage on the capabilities of LSTM-based networks to model long-term dependencies, a sliding time window processing technique was adopted to generate sequence data from the raw multi-feature time series data. A fixed length (= seq_len) time window was utilized to sample multivariate data points from the training

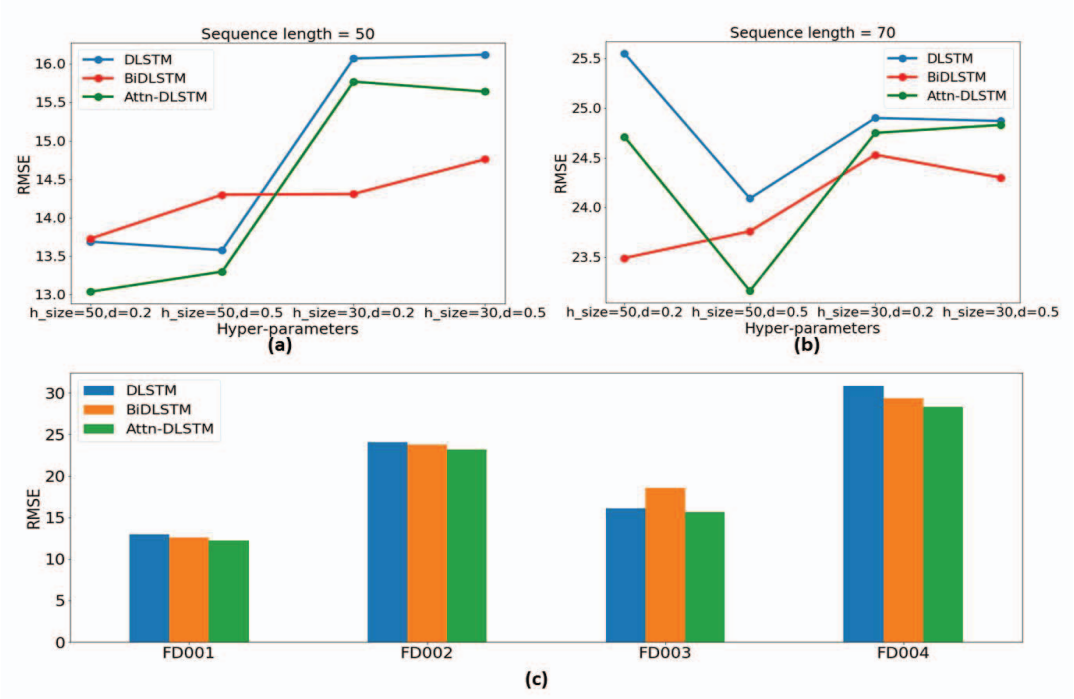

Fig. 5: Performance of different combinations of hyper-parameters. (a) RMSE for FD001 using seq_len = 50. (b) RMSE for FD002 using seq_len = 70. (c) RMSE values for all four datasets using seq_len = 70, h_size = 50 and d = 0.5.

datasets at consecutive cycles and then time window was shifted by one cycle to generate a new sample. This process was repeated up to the end-of-life of the engine to yield multiple multivariate time series with fixed sequence length, which were used as inputs to train the deep LSTM-based models used in this work. For test data, one multivariate sample of fixed length (= seq_len) was generated by taking the data from last consecutive cycles.

C. Results and Discussions

In this section, we present the results of RUL prediction for the four C-MAPSS sub-datasets: FD001, FD002, FD003 and FD004 using our proposed recurrent architectures in Section II. Next, we compare and contrast the performance of the proposed algorithms and understand the impact of attention.

1) Choice of hyper-parameters: We analyzed the effects of varying network structure on RUL prediction. For this purpose we used different values of number of hidden units per hidden layer (h_size = 30, 50), dropout rate (d = 0.2, 0.5) and sequence length (seq_len = 50, 70). Fig. 5(a) shows the RMSE values for FD001 for seq_len = 50 for different values of h_size and d for the three methods. We can see that the performance of attention based DLSTM network is always better than that of DLSTM network. Similar trend is seen in Fig. 5(b), which shows the RMSE value for FD002 for seq_len = 70. Moreover, the performance of BiDLSTM network was seen to be mostly better than that of DLSTM, whereas as compared with the attention based DLSTM network, BiDLSTM's performance varies with different combinations of hyper-parameters. Based on this analysis, we selected the h_size value of 50 to provide more flexibility to the network to learn and d = 0.5 to prevent overfitting. Moreover, this combination of h_size and d values yields lower RMSE values for both values of seq_len considered and was therefore used for the rest of experiments in this work. Fig. 5(c) shows the performance comparison of DLSTM, BiDLSTM and Attn-DLSTM for all four datasets using seq_len = 70, h_size = 50 and d = 0.5.
2) Analysis of attention weights: We analyzed the attention weights learned by our proposed method Attn-DLSTM. Fig. 6 shows the attention weights α_t defined in equation (7) for all LSTM cells corresponding to seq_len = 70. The learned weights highlight and reinforce our hypothesis that RUL prediction depends not only on the last time step but also on other time steps. It can be clearly seen from Fig. 6 that different temporal sub-sequences play role in the prediction process. Moreover, the attention weights for the C-MAPSS dataset FD001 and FD003 are higher for about first 10-20 LSTM cells or time cycles while the attentional focus for FD002 and FD004 is more on the latter part of the sequence, which also points to the dataset similarities in terms of both FD001 and FD003 having data from only one operating condition and both FD002 and FD004 with six operating conditions.
3) Comparison with other reported results: We compared

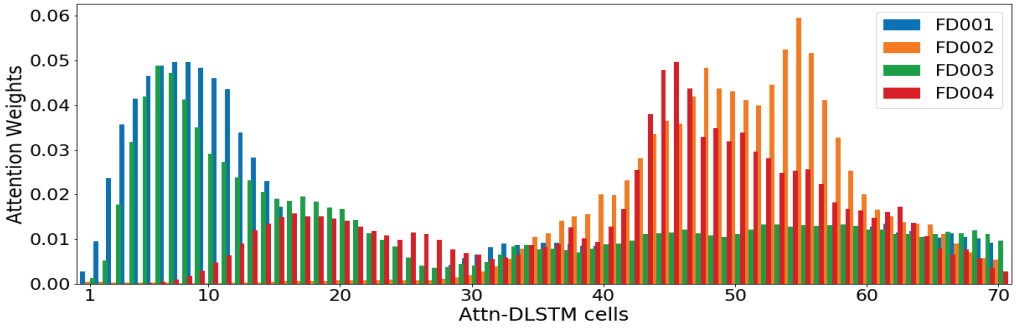

Fig. 6: Attention weights learned by the proposed Attn-DLSTM network for four C-MAPSS sub-datasets.

Methods	FD001	FD002	FD003	FD004
MLP [6]	37.56	80.03	37.39	77.37
SVR [6]	20.96	42.00	21.05	45.35
RVR [6]	23.80	31.30	22.37	34.34
CNN [6]	18.45	30.29	19.82	29.16
DLSTM	12.93	24.09	16.09	30.86
BiDLSTM	12.58	23.76	18.58	29.37
Attn-DLSTM	**12.22**	**23.16**	**15.61**	**28.30**

TABLE II: Performance comparisons of Attn-DLSTM with other methods on C-MAPSS datasets

the performance of our presented methods with the results generated in [6]. Table II shows the RMSE values for the four datasets as reported in [6] and the RMSE values obtained using the three methods in our work (in bold). The comparison shows that Attn-DLSTM performs the best for all datasets for the selected combination of network hyper-parameters seq_len, h_size and d.

IV. CONCLUSIONS

RUL prediction is a key problem in manufacturing domain. This paper presents three different architectures for estimation of RUL of turbofan engines. We first presented DLSTM, a two-layer LSTM-based network with dropout for the prediction task. Next, we improved upon the network by employing a bidirectional LSTM (BiDLSTM) which has the capability of taking into account both past and future data while prediction. Further, we employed the attention mechanism to develop an attention based deep LSTM network (Attn-DLSTM) to improve the prediction by emphasizing on other time steps in addition to the final time step. It was found that the Attn-DLSTM network performs better than DLSTM and BiDLSTM in most of the cases. Incorporation of attention in deep networks also increases the interpretability of the network. We also implemented other variants of deep recurrent architectures with varying number of fully connected layers, different dropout conditions, different number of hidden neuron in each layer and other activation functions like ReLU. Experiments showed that the architecture presented in the paper performs better than these variants in general. Comparison with other algorithms like SVR, MLP, RVR and CNN show the efficacy and usefulness of the presented deep recurrent architectures in this paper.

REFERENCES

[1] K. Hosseinkhani and E. Ng, "A combined empirical and numerical approach for tool wear prediction in machining," *Procedia CIRP*, vol. 31, pp. 304–309, 2015.

[2] A. J. Torabi, M. J. Er, X. Li, B. S. Lim, and G. O. Peen, "Application of clustering methods for online tool condition monitoring and fault diagnosis in high-speed milling processes," *IEEE Systems Journal*, vol. 10, no. 2, pp. 721–732, 2016.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] R. Socher, Y. Bengio, and C. D. Manning, "Deep learning for nlp (without magic)," in *Tutorial Abstracts of ACL 2012*. Association for Computational Linguistics, 2012, pp. 5–5.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] 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 applications*. Springer, 2016, pp. 214–228.

[7] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

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

[9] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, 2017, pp. 88–95.

[10] C.-G. Huang, H.-Z. Huang, and Y.-F. Li, "A bi-directional lstm prognostics method under multiple operational conditions," *IEEE Transactions on Industrial Electronics*, 2019.

[11] F. Ma, R. Chitta, J. Zhou, Q. You, T. Sun, and J. Gao, "Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 1903–1911.

[12] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems*, 2016, pp. 3504–3512.

[13] E. Ramasso and A. Saxena, "Performance benchmarking and analysis of prognostic methods for cmapss datasets." *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15, 2014.

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR, abs/1211.5063*, vol. 2, 2012.

[15] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.