SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

LSTM Inefficiency in Long-Term Dependencies Regression Problems

Safwan Mahmood Al-Selwi[1,2,*], Mohd Fadzil Hassan[2], Said Jadid Abdulkadir[1,2], Amgad Muneer[1,3]

[1] Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, 32610 Seri Iskandar, Perak Darul Ridzuan, Malaysia
[2] Center for Research in Data Science (CeRDaS), Universiti Teknologi PETRONAS, 32610 Seri Iskandar, Perak Darul Ridzuan, Malaysia
[3] Department of Imaging Physics, The University of Texas MD Anderson Cancer Center, Houston, TX 77030, USA

ARTICLE INFO

Article history:
Received 1 December 2022
Received in revised form 29 March 2023
Accepted 10 April 2023
Available online 2 May 2023

Keywords:
Recurrent Neural Networks; regression problems; Vanishing Gradient Problem; Long Short-Term Memory; long-term dependencies

ABSTRACT

Recurrent neural networks (RNNs) are an excellent fit for regression problems where sequential data are the norm since their recurrent internal structure can analyse and process data for long. However, RNNs are prone to the phenomenal vanishing gradient problem (VGP) that causes the network to stop learning and generate poor prediction accuracy, especially in long-term dependencies. Originally, gated units such as long short-term memory (LSTM) and gated recurrent unit (GRU) were created to address this problem. However, VGP was and still is an unsolved problem, even in gated units. This problem occurs during the backpropagation process when the recurrent network weights tend to vanishingly reduce and hinder the network from learning the correlation between temporally distant events (long-term dependencies), that results in slow or no network convergence. This study aims to provide an empirical analysis of LSTM networks with an emphasis on inefficiency in long-term dependencies convergence because of VGP. Case studies on NASA's turbofan engine degradation are examined and empirically analysed.

1. Introduction

A recurrent neural network (RNN) is a model of neural network for modelling time-series data devised in the 1980s [1,3]. Through the connections between hidden units linked with the time delay, the network can preserve information about the past, allowing it to uncover temporal correlations between events that occurred in the data at long distances apart. Even though the core function of RNNs is to learn long-term dependencies (the temporal correlations or dependencies between events far away from each other), theoretical and experimental evidence indicates that learning to process information for a long time is challenging. One solution to this issue is to add explicit memory to a network. In 1997, Hochreiter and Schmidhuber [4] created the first suggestion of this type, the long short-term memory (LSTM) with particular hidden units whose natural tendency is to retain inputs for a longer period of time. Later in 2014, the gated recurrent unit (GRU) was developed by Cho *et al.,* [5] to make it possible for each recurrent unit to capture the relationships between

* *Corresponding author.*
E-mail address: safwan_21002827@utp.edu.my

https://doi.org/10.37934/araset.30.3.1631

differences on different time scales adaptively. Like the LSTM unit, the GRU unit features gating mechanisms that regulate the information flow without the need for separate memory cells [6].

RNNs are an excellent fit for regression problems where sequential data are the norm due to the fact that their recurrent internal structure can analyze and process data for long. However, two widely known issues with training RNNs are the exploding and the vanishing gradient [6]. The exploding gradient problem (EGP) happens when long-term elements exponentially grow more than short-term dependencies. On the other hand, the vanishing gradient problem (VGP) happens when long-term elements grow exponentially and rapidly to norm 0, preventing the model from learning the correlation between temporally distant events [7] It is due to the vanishingly small value of the gradients, which causes slight or negligible improvement in weights. A deep learning model may require more time to train and learn from the data, indicating that there is little or no neural network convergence [8].

Based on experimental findings, gated units, such as LSTM and GRU, outperform the Vanilla RNNs in terms of prediction and validation accuracy. Numerous recent studies stated that LSTM could mitigate or lessen the VGP through effective learning [9-14]. Although gated units were originally developed to solve this problem, this problem was, and still is, an unsolved problem and gated units can still run or face this issue especially for long-term inputs and outputs dependencies, but not nearly as much as the Vanilla RNN [15-17].

Furthermore, Bayer [18] demonstrates that LSTM can still encounter VGP because of the gradients given in the below Eq. (6), where repeated multiplications of recurrent weights U can lead the gradients to vanish, making it difficult to calculate long-term dependencies. Consequently, LSTM may continue to struggle to identify long-term dependencies in excessively lengthy sequences. Besides, according to Chandar *et al.,* [19], the gradients on the gating units themselves vanish as the units' activations functions saturate and cause the problem. From this observation, they introduced the Non-saturating Recurrent Units (NRUs) to mitigate the problem of vanishing gradient further.

To analyze LSTM models used in regression problems, we have selected the well-known public data set for asset degradation modelling generated by NASA's C-MAPSS software [20,21] for the literature comparison in this study. C-MAPSS is the benchmark data set for remaining useful life (RUL) regression problems [22]. The data set is distributed into four different simulated subsets (FD001 to FD004), provides multivariate time series data with various operational modes, and contains run-to-failure simulated data from 218 turbofan jet engines whose readings are measured by 21 attached sensors. Root mean squared error (RMSE) is used as an evaluation metric since it is the most applicable evaluation metric for RUL prognosis models, according to Vollert *et al.*, [23].

The rest of this paper is organized as follows: Section 2 highlights the research background. Section 3 presents some related works to regression problems in LSTM, along with a deep discussion and results. Section 4 shows our experiment about how efficient LSTM is in handling long-term dependencies. Finally, this study is summarized in section 5.

2. Background

This section provides a background about the research problem in three sub-sections. The RNN overall training process is highlighted in sub-section 2.1. The vanishing gradient problem is discussed in sub-section 2.2. The LSTM cell structure and operations are covered in sub-section 2.3.

2.1 RNN Overall Training Process

The overall training process of RNN consists of two passes, a forward pass, and a backward pass, and it is illustrated in Figure 1 [16].

i. Compute the "forward pass" equations using initialized weights/hidden states.
ii. Compute and evaluate the "loss/error" function using the given ground truth targets/labels and the predicted values/labels as shown in this equation

$$J(W,b) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}^i, y^i)^2 \quad (1)$$

where *W* denotes the weight matrix, *b* denotes the bias vector, *n* denotes the number of inputs, $\hat{y}^i$ denotes the predicted value, y^i denotes the true observed value, and *L* denotes the loss value between the predicted and the true observed value. The cost function *J*, which is the square of the difference between the true observed value and the predicted value, can be used to figure out how much this loss is.

i. Compute the "backward pass" equations to acquire the required gradients to update the current weights with the goal of reducing (optimizing) the loss/error function.
ii. Update the weights using the acquired gradients and repeat steps (a and b).
iii. Repeat steps (a to d) until a satisfactory loss/error threshold value is reached.

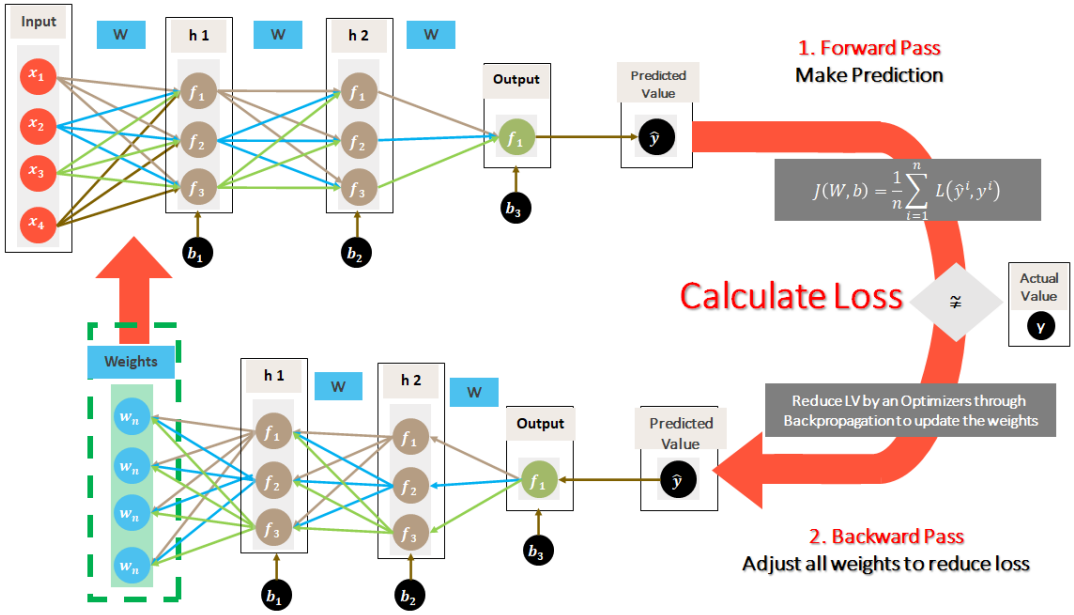

Fig. 1. Recurrent neural network overall training process

2.2 Vanishing Gradient Problem (VGP)

RNN operates by the fundamental of storing the output of a layer and sending it back to the input to predict the output of the layer. However, RNNs suffer from a remarkable problem known as vanishing gradient or gradient disappearance, particularly when learning long-term dependencies (usually within 10:15 time steps) [24]. Hochreiter and Schmidhuber [4] first discovered this problem

in 1991. The primary reason for this phenomenon is the decrease in gradients during backpropagation epochs while updating the weights of the network's prior layers [7,15]. It occurs when the gradients become exponentially smaller and approach 0 during the training process. They no longer contribute to finding the optimum weights needed to learn long-term data dependencies. This problem makes the RNN unable to bridge the long-term dependencies within the data due to infinitesimally small gradients in the previous time steps.

RNN updates its hidden state mathematically as follows by taking X_t as input at any time step *t*.

$$S_t = Wx_t + Uh_{t-1} \tag{2}$$

$$h_t = f(S_t) \tag{3}$$

where *W* denotes the input weight, *U* denotes the recurrent weight of the network, and f denotes a non-linear activation function like *tanh* or *sigmoid*. Consider a T-length sequence with a loss of $\mathcal{L}$ determined when the sequence is over. To calculate the gradient of the loss function $\frac{\partial \mathcal{L}}{\partial U_{ij}}$ at time t, we must first calculate $\frac{\partial \mathcal{L}}{\partial h_t}$ using this chain rule

$$\frac{\partial \mathcal{L}}{\partial h_t} = \frac{\partial \mathcal{L}}{\partial h_T} \frac{\partial h_T}{\partial h_t} \tag{4}$$

$$= \frac{\partial \mathcal{L}}{\partial h_T} \prod_{k=t}^{T-1} \frac{\partial h_{k+1}}{\partial h_k} \tag{5}$$

$$= \frac{\partial \mathcal{L}}{\partial h_T} \prod_{k=t}^{T-1} (\text{diag}\,[f'(S_k)]U) \tag{6}$$

For a very long sequence length T, multiplying *U* over and over in the last Eq. (6) during the backpropagation through time (BPTT) process can cause the network gradients to grow or shrink at an exponential rate causing the problem of exploding or vanishing gradient respectively.

Another cause of VGP in deep RNN is the use of saturated activation functions f, such as the logistic function (sigmoid) σ (·) or a hyperbolic tangent (tanh) (.) [19]. The difference between the variance of these activation functions' inputs and outputs is extremely large because they transform and shrink a bigger input space into a smaller output space that is between (0, 0.25) in sigmoid and between (0,1) in tanh [25]. Also, using the normalized exponential function (softmax) can leave the network susceptible to vanishing gradient.

Although RNNs improve predictions in many regression tasks, vanishing and exploding gradient issues are more likely to occur when the network is deeper, which exacerbates the problem. Moreover, VGP makes it challenging to determine in which direction the cost function parameters should be adjusted to increase its performance. In RNNs, only a portion of the gradients at time steps far enough from the current input may vanish, making diagnosing the problem difficult. Since information travels through time and layers in the RNN, and the weights are shared across all time steps, examples with longer sequences can frequently come across this problem. In the worst-case scenario, if all the gradient terms in BPTT equations vanish or explode, the network stops learning altogether; therefore, further improvement of the network performance becomes impossible without changing the hyperparameters of the network. One way to diagnose both issues is to track the norm of the temporal components of the gradients. If the norm at a given time step is

substantially lower than 1, the vanishing gradient issue is present [16]. Conversely, if it is substantially higher than 1, then the exploding gradients issue is present.

Nonetheless, with LSTM, the gradients will not exponentially decay as it does with Vanilla RNN. There is at least one optimal approach to a combination of weights that mitigates the VGP as much as possible and avoids the training stagnating. Bayer's [18] observation regarding this issue in LSTM demonstrates the importance of initializing and optimizing the network's weights using gradient descent or metaheuristic algorithms enhancing the model architecture, applying batch normalization, choosing efficient activation functions or using non-saturating activation functions such as rectified linear unit (ReLU) as all of these can help avoid vanishing gradient for longer periods in LSTM [8,26-30] Thus, enabling more efficient learning of long-term dependencies. However, even with these techniques, the ability of an LSTM to remember long-term dependencies will still be limited by the specific architecture and training of the model, the data it is being applied to, and the performance metrics being used to evaluate it [16,18].

2.3 Long Short-Term Memory

Neural network topologies such as RNNs are an excellent fit for regression problems where sensors' sequential data are the norm because their internal structure can analyze and process sequential input data [4]. However, because of the VGP, RNNs have difficulty learning long-term dependencies. In 1997, Hochreiter and Schmidhuber [4] devised the long short-term memory (LSTM) with special hidden units whose natural tendency is to remember inputs for a longer period of time and minimize the likelihood of VGPs in RNNs. With the invention of LSTM, two main terms were added to the simple RNN terminologies, the cell states, and the gates. The cell states can be thought of as a storage or long-term memory that stores past relevant information while the gates manage and regulate the flow of information through the network. Besides, LSTM has also hidden states and they are known as short-term memory. Therefore, many time series applications can benefit from implementing LSTM such as network latency reduction in healthcare, autonomous negotiation, web services [31-33] It also can be used in detection tasks such as Iron oxide nanoparticles, fatigue crack growth prediction and gas coning detection [34-36].

LSTM may be presented as an RNN with a memory pool and two key vectors

i. Short-term state that maintains the current time-step's output.
ii. Long-term state that handles long-term entities while travelling through the network.

Both the cell states and the hidden states flow forward through time while getting updated and influencing each other at each time step. Unlike the hidden state, the cell state is usually not used as an output, but it is passed and updated internally along with the hidden states at each time step and influences the hidden state at each time step. Additionally, the flow of information in LSTM networks is regulated by three internal gates during the learning process: the input gate, the output gate, and the forget gate. The LSTM cell structure and operations are highlighted in the sub-sections below.

2.3.1 LSTM cell structure

The generalized single-cell LSTM structure with its gates is illustrated in Figure 2, and the LSTM cell architecture is illustrated in Figure 3.

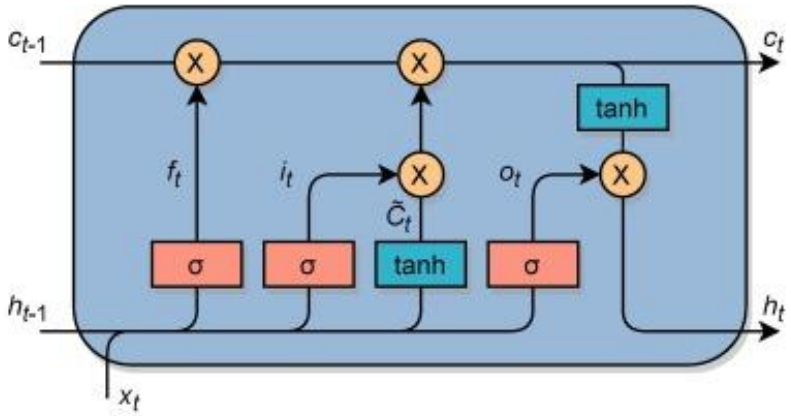

Fig. 2. A generalized single cell LSTM structure [37]

Where the input gate i_t controls what relevant long-term information can be added to the cell state c_t from the new candidate state vector $\tilde{c}_t$. The forget gate f_t, was introduced in 1999 by Gers *et al.,* [38], eliminates irrelevant data with less significance on the prediction progress to allow new information to be processed in the cell state. Finally, the output gate o_t outputs the valuable data from the current cell state.

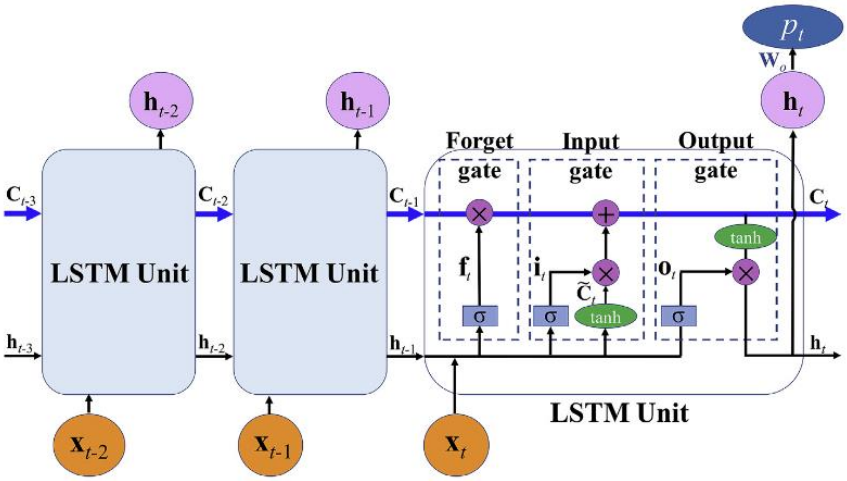

Fig. 3. LSTM architecture [10]

2.3.2 Single cell LSTM operations

The internal operations and activities of a single cell LSTM are explained in Algorithm 1, where σ denotes the logistic sigmoid function, *tanh* denotes the hyperbolic tangent function, and $\odot$ denotes the Hadamard product (element-wise product) [4,37].

Algorithm 1
Single cell LSTM operations

Given	W	Matrix of input-to-hidden weights
	U	Matrix of hidden-to-hidden weights
	b	Vector of Bias
Input	x_t	Vector of input at time step t
	c_{t-1}	Vector of previous memory cell state
	h_{t-1}	Vector of previous hidden state
Output	c_t	Vector of memory cell state
	h_t	Vector of hidden state

Process

1. Compute the input gate vector

$$\boldsymbol{i_t = \sigma\left(W^{(i)}\, x_t + U^{(i)}\, h_{t-1} + b^{(i)}\right)} \quad (7)$$

2. Compute the forget gate vector

$$\boldsymbol{f_t = \sigma\left(W^{(f)}\, x_t + U^{(f)}\, h_{t-1} + b^{(f)}\right)} \quad (8)$$

3. Compute the output gate vector

$$\boldsymbol{o_t = \sigma\left(W^{(0)}\, x_t + U^{(0)}\, h_{t-1} + b^{(0)}\right)} \quad (9)$$

4. Compute the memory cell state vector

$$\boldsymbol{c_t = i_t \odot u_t + f_t \odot c_{t-1}} \quad (10)$$

where

$$\boldsymbol{u_t = tanh\left(W^{(u)}\, x_t + U^{(u)}\, h_{t-1} + b^{(u)}\right)} \quad (11)$$

5. Compute the hidden state vector

$$\boldsymbol{h_t = o_t \odot tanh(c_t)} \quad (12)$$

Based on experimental findings, gated units outperform the Vanilla RNNs in terms of prediction and validation accuracy. Still, they have four times as many parameters as the Vanilla RNN, and hence they suffer from high complexity in the hidden layers [26]. LSTM has more trainable weights than Vanilla RNN. Specifically, it features four distinct sets of trainable input weights for input data: $[W_f, W_i, W_o, W_c]$ and four distinct sets of trainable recurrent weights for hidden states: $[U_f, U_i, U_o, U_c]$. In addition, there are also four distinct trainable bias terms: $[b_f, b_i, b_o, b_c]$.

3. Results and Discussion

This section presents some related works to regression problems in LSTM, along with a deep discussion and results. Vollert *et al.,* [23] published a review paper on the challenges of RUL prognosis using machine learning-based methods. The majority of the 81 reviewed papers were based on LSTM models. On the other hand, the C-MAPSS data set is the standard benchmark for evaluating RUL regression problems [22]. Thus, the case studies on NASA's turbofan engine degradation are examined and empirically analyzed. Table 1 shows the comparison of related works.

Table 1
Related works to LSTM networks used in remaining useful life prediction

Publication	Year	Techniques	Findings	Discussion
Bae *et al.,* [39]	2022	PHT - LSTM	Physical health timesteps (PHT) were introduced as an additional input feature to the proposed LSTM network to facilitate learning the relationship between sensory data and the remaining useful life. Utilizing the LSTM model to exploit the long-term dependencies of the temporal information in the time series data, this approach enhances the RUL prediction by refining the timestep information to be helpful.	Before estimating the RUL, they first estimate the training and test data PHT. The estimated PHT is then added to the input features for the primary goal of RUL estimation. However, this pre-calculation of the PHT can be time-consuming for systems that require rapid predictions. Besides, they used three distinct LSTM networks for 1) Estimating PHT for training units, 2) Estimating PHT for test units, and 3) Estimating RUL for the test units with the estimated PHT. Although the authors mentioned that the VGP could undermine their LSTM predictions in long-term dependencies, their study did not cover the problem of vanishing gradient and its impact on their LSTM models.
Muneer *et al.,* [40]	2021	LSTM with Attention Mechanism	They presented LSTM with an attention mechanism to depict the association between the selected features and the RUL. Due to the dimensionality reduction, their model generated better results in terms of improved prediction quality and increased computing performance.	This method could not capture the long-term dependencies entirely because of the VGP, and it was tested on the easiest subsets of the data set. Besides, the researchers used the softmax activation function in their LSTM model with 30 neurons, which leaves the network susceptible to vanishing gradient.
Xie *et al.,* [41]	2021	AM-ConvFGRNET	The proposed attention convolutional forget gate recurrent network (AM-ConvFGRNET) has two phases. The first one is a forget gate convolutional recurrent network (ConvFGRNET) that is proposed based on a one-dimensional analogue LSTM, where all gates except the forget gate are removed, and chrono-initialized biases are used. The second one is the attention mechanism, which assures the method captures more particular features in order to generate better results.	The proposed model takes longer time to capture long-term dependencies compared with others (more than eight minutes in the FD002 subset). Possible reasons for that could be using the raw vibration signals as input directly without removing the outliers. Besides, using the softmax activation function inside their LSTM model, which leaves the network susceptible to VGP.
Kumar [42]	2021	CNN-LSTM	A hybrid model composed of stacked CNN layers for features extraction and an LSTM layer to estimate the RUL. A genetic algorithm was used for hyperparameter selections and optimization.	The CNN-LSTM model was developed and compared with numerous models, such as HDNN, deep CNN, and deep LSTM. However, the results indicate that the HDNN model scored the best in multi-operating condition subsets, while the deep CNN model scored

				the best in single-operating condition subsets. i.e., this study gives low RUL predictions compared with other studies. Moreover, the computation time is extremely long (up to 6 hours in the FD002 subset), which proves that this LSTM model suffers from VGP and its consequences in long-term dependencies.
Zhao *et al.,* [43]	2020	A double-channel hybrid model based on bidirectional LSTM and CNN	The authors argued that both CNN and LSTM have limitations and drawbacks in RUL prediction. CNN is well known for its capabilities for extracting highly abstract features according to the spatial feature method, but it neglects the time sequence of the data. On the contrary, LSTM performs better with time-series data, but it is incapable of spatial data extraction. Thus, they proposed a model based on CNN and BiLSTM so that each method fulfils the gap of the other.	The hybrid model's first channel is composed of 3 layers of CNN with filter size (1*10), (1*3), and (1*3), respectively, to obtain the spatial features from the data, whereas the second channel has stacked 2 BiLSTM with 16 and 8 neurons respectively to obtain the long-term data dependencies. All these layers use the ReLU activation function, and the maximum training epoch was set as 200. However, this hybrid model did not completely capture the long-term dependencies, especially in the complex dataset FD002 and FD004.
Da Costa *et al.,* [44]	2019	LSTM network with a global attention mechanism	An LSTM network with a global attention mechanism was developed for learning RUL relationships directly from time-series data. To improve the model results, the researchers also focused on the hyperparameters analysis, such as learning rate, batch size, number of LSTM layers, number of neurons, and time window size. They also proposed a soft attention mechanism for learning attention weights at each RUL estimation timestep.	The authors mentioned that gated architectures reduce the VGP. They performed 10-fold cross-validation in order to determine the best network architecture. Then, they concluded that having one LSTM layer with 100 neurons resulted in the best performance, whereas adding more LSTM layers only added more complexity to the network without enhancing the performance.
Zheng *et al.,* [45]	2017	Deep LSTM	The deep model that was employed contains two LSTM layers, each with a different number of nodes (32 and 64), followed by an (8-8-1) neural network layer. They control the model overfitting by utilizing dropout and L2 regularization in addition to the RMSprop optimizer while training models.	This study gives low RUL prediction results compared with other studies in this paper, making it impractical for real prognosis problems. Early stopping was utilized to stop the learning process when there was no improvement in the data validation.

The findings and discussion of the literature reviewed are highlighted in Table 1 The authors have used different architectures, techniques, and hyperparameters tunings to enhance their models and to get more reliable and accurate RUL predictions. The RMSE results across the surveyed models, and the four subsets of the C-MAPSS dataset (FD001 to FD004) are listed in Table 2 and plotted in Figure 4.

Table 2
RMSE results across models and subsets

Techniques	RMSE			
	FD001	FD002	FD003	FD004
PHT – LSTM [39]	9.75	12.90	12.16	23.10
LSTM with Attention Mechanism [40]	-	12.87	11.23	-
AM-ConvFGRNET [41]	12.67	16.19	12.82	19.15
CNN-LSTM [42]	15.68	22.26	16.89	26.32
A double-channel hybrid model based on bidirectional LSTM and CNN [43]	12.58	19.34	12.18	20.03
LSTM network with a global attention mechanism [44]	13.95	17.65	12.72	20.21
Deep LSTM [45]	16.14	24.49	16.18	28.17

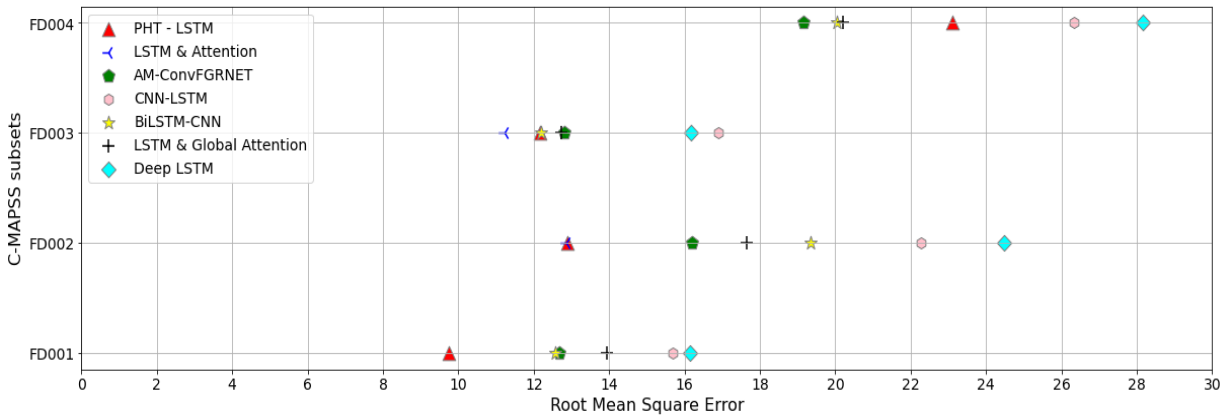

Fig. 4. Comparison of RMSE results across models and subsets

It is clearly shown that the best result in RMSE among the literature surveyed is (9.75) by Bae *et al.,* [39] for the subset FD001. On the other hand, the worst result is (28.17) generated by the deep LSTM model designed by Zheng *et al.,* [45] for the subset FD004. Besides, the LSTM model with attention mechanism designed by Muneer *et al.,* [40] scored (12.87) and (11.23) for the subset FD002 and FD003, respectively. However, their models were not tested with the most complicated subset in C-MAPSS, the FD004 subset. In that subset, the AM-ConvFGRNET model based on LSTM and proposed by Xie *et al.,* [41] generated a value of (19.15).

Researchers proposed numerous LSTM models with different techniques to mitigate VGP and EGP as much as possible and to obtain better results from their designed models, such as

i. Applying weights regularization to prevent the gradients from becoming too small and to encourage the model to learn more stable and generalizable patterns in the data.
ii. Building deeper LSTM architectures with multiple layers to help their models to learn more complex dependencies and to better retain information over longer sequences of inputs.
iii. Initializing and optimizing the models' weights using gradient descent or metaheuristic algorithms.
iv. Testing and trying different activation functions.
v. Tuning the model's hyperparameters such as learning rate, number of LSTM layers, batch size, number of neurons, and time window size.

However, even with these solutions, the ability of LSTM models to remember long-term dependencies will still be limited by the specific architecture and training of the model, the data it is being applied to, and the performance metrics being used to evaluate these models [16,18].

To prove the inefficiency of LSTM on long-term dependencies, an experiment was developed using Python programming language for building a sequential LSTM model. The idea is to train the LSTM model on a sequence of numbers where the value of each number is determined by the previous number, with a long-time gap between the relevant numbers to analyze the long-term dependencies among the data. First, the length of the sequence (N) and the gap between relevant numbers (G) are initialized. Then, the training and testing data are generated by creating two lists of numbers from 1 to N, with some of the numbers changed according to the gap value. After that, the developed LSTM model is trained for (100) epochs on the training data that have some missing values. Finally, the model is evaluated on the testing data with the same missing values.

This section includes three sub-sections; the experiment setup and model configuration in sub-section 3.1, the evaluation metrics used to evaluate the developed model in sub-section 3.2, the experiment results and discussion in sub-section 3.3.

3.1 Experiment Setup

The developed LSTM sequential model consists of two layers: an LSTM layer and a dense layer. The LSTM layer has 32 units and expects input with any number of time steps, where each time step has one feature. The dense layer has one unit, which is the model's output.

The experiment is performed on Intel (R) Core (TM) (i7-5600U) CPU @ 2.60GHz, (16.0) GB DDR3 RAM using Python (3.9.12), Anaconda (22.11.1), TensorFlow (2.11.0). Besides, the following libraries: Pandas (1.5.2), NumPy (1.23.5), Matplotlib (3.5.3), and Keras (2.11.0).

3.2 Experiment Evaluation Metrics

Four evaluation metrics are used to evaluate the developed LSTM model: Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). These metrics are defined as

3.2.1 Mean absolute error (MAE)

It represents the average of the absolute difference between the actual and predicted values in the dataset, and it is denoted as

$$\frac{1}{n}\sum_{1}^{n} |y_i - \hat{y}_i| \tag{13}$$

3.2.2 Mean squared error (MSE)

It represents the average of the squared difference between the original and predicted values in the dataset, and it is denoted as

$$\frac{1}{n}\sum_{i}^{n} (y_i - \hat{y}_i)^2 \tag{14}$$

3.2.3 Mean absolute percentage error (MAPE)

It is a measure of the prediction accuracy of a forecasting method in statistics, and it is denoted as

$$\frac{1}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{y_t} \times 100 \quad (15)$$

3.2.4 Root mean squared error (RMSE)

It is defined as the standard deviation of the RUL prediction errors, and it measures how far data points are plotted from the regression line, and it is denoted as

$$\sqrt{\frac{1}{\mathrm{n}}\sum_{\mathrm{j}=1}^{\mathrm{n}}\left(\mathrm{y_j} - \hat{\mathrm{y}}_\mathrm{j}\right)^2} \quad (16)$$

where n represents the total number of predictions made, y_j is the true observed value, and $\hat{y}_j$ represents the predicted value. When both values are equal, the plotted points lie on the regression line, so no prediction errors occur. When the predictions errors increase, the values of the RMSE also increase correspondingly.

3.3 Experiment Results and Discussion

This experiment is repeated multiple times to test the model on different sequence lengths (10, 20, 30, 40, 50, 60, 70, 80, 90, and 100), where the gap between the relevant numbers is set as (3). In each sequence length, the model is evaluated, and the metrics are stored and plotted. In conclusion, when tested on a longer sequence with a larger gap, the model had higher metrics values, indicating that it is less efficient in making predictions for long-term dependencies. The evaluation metrics values over an increased sequence length are shown in Table 3 and are plotted in Figure 5.

Table 3
LSTM model loss over sequence length

Metrics / Sequence Length	MAE	MSE	MAPE	RMSE
10	2.32808	7.291632	56.92557	2.700302
20	6.750084	63.25161	59.2161	7.953088
30	12.46396	206.8792	67.89247	14.3833
40	18.97133	467.5989	74.83435	21.62403
50	25.95008	855.7687	81.26044	29.25352
60	31.18816	1230.381	82.7712	35.07679
70	37.78834	1789.665	86.09274	42.30444
80	43.95028	2412.183	87.8178	49.11398
90	47.4785	2843.848	85.13817	53.32774
100	55.66957	3848.653	90.10715	62.03751

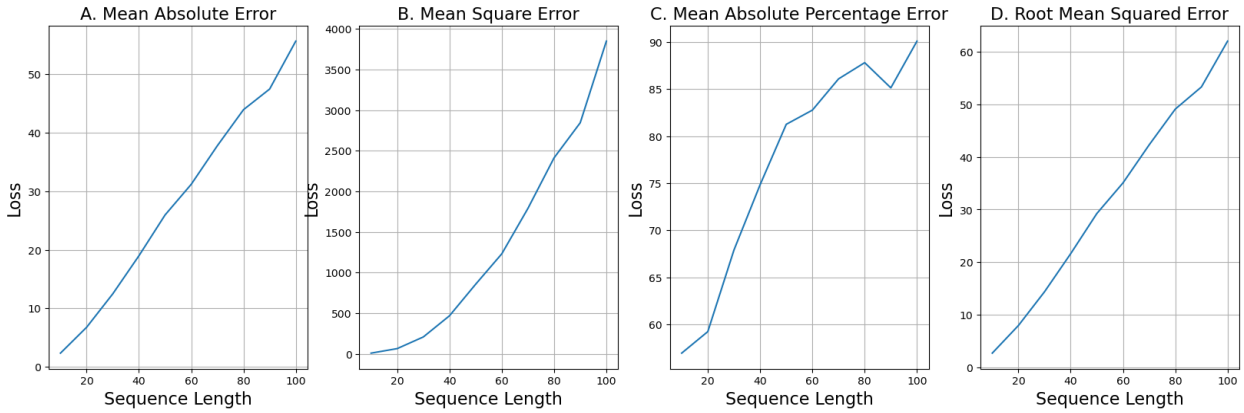

Fig. 5. LSTM model loss over sequence length

Moreover, the length of the input sequence can affect the performance of the LSTM model in several ways. One way is by determining the amount of context the model has made available when making predictions. LSTM models can remember long-term dependencies in the data, but the longer the input sequence, the harder it is for the model to remember all the relevant information. If the sequence length is too short, the model may not have enough context to make accurate predictions, but if the sequence length is too long, the model may have difficulty retaining the necessary information. Finding the optimal sequence length for a given dataset and prediction task is often a matter of experimentation and tuning.

Another way the sequence length can affect model performance is by determining the amount of training data needed to train the model. LSTM models may require a large amount of training data, and the longer the input sequence, the more training data is needed to fully capture the data dependencies. If the available training data is insufficient to train the model adequately, the model's performance may be poor.

4. Conclusions

RNNs, with their recurrent internal structure, can analyze and process sequential data in regression problems for a long time. However, RNNs are prone to the problems of exploding and vanishing gradients. Although gated units, such as LSTM and GRU, were initially created to solve these problems, these problems remain unsolved. Gated units facing these problems, especially for long-term inputs and outputs dependencies. Nonetheless, with LSTM, the gradients do not exponentially degrade as they do with Vanilla RNN.

Case studies on NASA's turbofan engine degradation are examined and empirically analyzed to understand how VGP affects LSTM performance. Besides, to prove the inefficiency of LSTM on long-term dependencies, an experiment is developed by using Python for a sequential LSTM model. The results showed that the longer the input sequence, the harder it is for the LSTM model to remember all the relevant information. Our future proposed model will be based on avoiding all causes of VGP and proposing an optimized metaheuristic method for LSTM weight optimization. We believe that there exists at least one ideal weight combination that mitigates the VGP as much as possible, prevents training stagnation, and improves LSTM performance in solving the regression problems.

Acknowledgement

Universiti Teknologi PETRONAS fully support this research under the Yayasan Universiti Teknologi PETRONAS Fundamental Research Grant (YUTP-FRG 1/2021 (015LC0-370).

References

[1] Werbos, Paul J. "Generalization of backpropagation with application to a recurrent gas market model." *Neural networks* 1, no. 4 (1988): 339-356. https://doi.org/10.1016/0893-6080(88)90007-X
[2] Jeffrey, L. Elman. "Finding structure in time." *Cognitive science* 14, no. 2 (1990): 179-211. https://doi.org/10.1207/s15516709cog1402_1
[3] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323, no. 6088 (1986): 533-536. https://doi.org/10.1038/323533a0
[4] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
[5] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014). https://doi.org/10.3115/v1/D14-1179
[6] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks* 5, no. 2 (1994): 157-166. https://doi.org/10.1109/72.279181
[7] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." In *International conference on machine learning*, pp. 1310-1318. Pmlr, 2013.
[8] Alqushaibi, Alawi, Said Jadid Abdulkadir, Helmi Md Rais, Qasem Al-Tashi, Mohammed G. Ragab, and Hitham Alhussian. "Enhanced weight-optimized recurrent neural networks based on sine cosine algorithm for wave height prediction." *Journal of Marine Science and Engineering* 9, no. 5 (2021): 524. https://doi.org/10.3390/jmse9050524
[9] Khademi, Zahra, Farideh Ebrahimi, and Hussain Montazery Kordy. "A transfer learning-based CNN and LSTM hybrid deep learning model to classify motor imagery EEG signals." *Computers in biology and medicine* 143 (2022): 105288. https://doi.org/10.1016/j.compbiomed.2022.105288
[10] Wei, Xin, Lulu Zhang, Hao-Qing Yang, Limin Zhang, and Yang-Ping Yao. "Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks." *Geoscience Frontiers* 12, no. 1 (2021): 453-467. https://doi.org/10.1016/j.gsf.2020.04.011
[11] Landi, Federico, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. "Working memory connections for LSTM." *Neural Networks* 144 (2021): 334-341. https://doi.org/10.1016/j.neunet.2021.08.030
[12] Ibrahim, Mariam, Ahmad Alsheikh, Qays Al-Hindawi, Sameer Al-Dahidi, and Hisham ElMoaqet. "Short-time wind speed forecast using artificial learning-based algorithms." *Computational Intelligence and Neuroscience* 2020 (2020). https://doi.org/10.1155/2020/8439719
[13] ElSaid, AbdElRahman, Fatima El Jamiy, James Higgins, Brandon Wild, and Travis Desell. "Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration." *Applied Soft Computing* 73 (2018): 969-991. https://doi.org/10.1016/j.asoc.2018.09.013
[14] Noh, Seol-Hyun. "Analysis of gradient vanishing of RNNs and performance comparison." *Information* 12, no. 11 (2021): 442. https://doi.org/10.3390/info12110442
[15] Rehmer, Alexander, and Andreas Kroll. "On the vanishing and exploding gradient problem in Gated Recurrent Units." *IFAC-PapersOnLine* 53, no. 2 (2020): 1243-1248. https://doi.org/10.1016/j.ifacol.2020.12.1342
[16] Kocoglu, Y., and S. Gorell. "Viable solutions to overcome weaknesses of deep learning applications in production forecasting: A comprehensive review." In *Unconventional Resources Technology Conference, 20–22 June 2022*, pp. 3279-3326. Unconventional Resources Technology Conference (URTeC), 2022. https://doi.org/10.15530/urtec-2022-3721904
[17] Eskandari, Hosein, Maryam Imani, and Mohsen Parsa Moghaddam. "Convolutional and recurrent neural network based model for short-term load forecasting." *Electric Power Systems Research* 195 (2021): 107173. https://doi.org/10.1016/j.epsr.2021.107173
[18] Bayer, J. S., "Learning sequence representations," Technische Universität München, 2015. Accessed: Apr. 08, 2022. [Online].
[19] Chandar, Sarath, Chinnadhurai Sankar, Eugene Vorontsov, Samira Ebrahimi Kahou, and Yoshua Bengio. "Towards non-saturating recurrent units for modelling long-term dependencies." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3280-3287. 2019. https://doi.org/10.1609/aaai.v33i01.33013280
[20] K. G. A. Saxena, "C-MAPSS Data Set, NASA Ames Prognostics Data Repository," 2008, Accessed: Apr. 15, 2022. [Online].
[21] D. K. Frederick, J. A. Decastro, and J. S. Litt, "User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)," 2007. [Online].
[22] Ferreira, Carlos, and Gil Gonçalves. "Remaining Useful Life prediction and challenges: A literature review on the use of Machine Learning Methods." *Journal of Manufacturing Systems* 63 (2022): 550-562. https://doi.org/10.1016/j.jmsy.2022.05.010

[23] Vollert, Simon, and Andreas Theissler. "Challenges of machine learning-based RUL prognosis: A review on NASA's C-MAPSS data set." In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1-8. IEEE, 2021. https://doi.org/10.1109/ETFA45728.2021.9613682
[24] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
[25] Tan, Hong Hui, and King Hann Lim. "Vanishing gradient mitigation with deep learning neural network optimization." In *2019 7th international conference on smart computing & communications (ICSCC)*, pp. 1-4. IEEE, 2019. https://doi.org/10.1109/ICSCC.2019.8843652
[26] Rashid, Tarik A., Polla Fattah, and Delan K. Awla. "Using accuracy measure for improving the training of LSTM with metaheuristic algorithms." *Procedia computer science* 140 (2018): 324-333. https://doi.org/10.1016/j.procs.2018.10.307
[27] Sharifi, Aboosaleh Mohammad, Kaveh Khalili Damghani, Farshid Abdi, and Soheila Sardar. "A hybrid model for predicting bitcoin price using machine learning and metaheuristic algorithms." *Journal of applied research on industrial engineering* 9, no. 1 (2022): 134-150. https://doi.org/10.22105/jarie.2021.291175.1343
[28] Lu, Peng, Lin Ye, Yongning Zhao, Binhua Dai, Ming Pei, and Yong Tang. "Review of meta-heuristic algorithms for wind power prediction: Methodologies, applications and challenges." *Applied Energy* 301 (2021): 117446. https://doi.org/10.1016/j.apenergy.2021.117446
[29] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In *International conference on machine learning*, pp. 448-456. pmlr, 2015.
[30] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807-814. 2010.
[31] Shukla, Saurabh, Mohd Fadzil Hassan, Low Tang Jung, Azlan Awang, and Muhammad Khalid Khan. "A 3-tier architecture for network latency reduction in healthcare internet-of-things using fog computing and machine learning." In *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, pp. 522-528. 2019. https://doi.org/10.1145/3316615.3318222
[32] Adnan, Muhamad Hariz Muhamad, Mohd Fadzil Hassan, Izzatdin Aziz, and Irving V. Paputungan. "Protocols for agent-based autonomous negotiations: a review." In *2016 3rd International Conference on Computer and information sciences (ICCOINS)*, pp. 622-626. IEEE, 2016. https://doi.org/10.1109/ICCOINS.2016.7783287
[33] Beer, Mohamed Ibrahim, and Mohd Fadzil Hassan. "Adaptive security architecture for protecting RESTful web services in enterprise computing environment." *Service Oriented Computing and Applications* 12, no. 2 (2018): 111-121. https://doi.org/10.1007/s11761-017-0221-1
[34] Kiew, Peck Loo, Nur Ainaa Mohd Fauzi, Shania Aufaa Firdiani, Man Kee Lam, Lian See Tan, and Wei Ming Yeoh. "Iron oxide nanoparticles derived from Chlorella vulgaris extract: Characterization and crystal violet photodegradation studies." *Progress in Energy and Environment* 24 (2023): 1-10. https://doi.org/10.37934/progee.24.1.110
[35] Venugopal, Arvinthan, Roslina Mohammad, and Md Fuad Shah Koslan. "Fatigue Crack Growth Prediction on Su-30MKM Horizontal Stabilizer Lug Using Static Analysis." *Journal of Advanced Research in Applied Mechanics* 99, no. 1 (2022): 10-23.
[36] Kanaani, Osamah Othman, Sami Abdelrahman Musa Yagoub, Shabir Habib, Akmal Aulia, and Bonavian Hasiholan. "Prediction of gas coning in hydrocarbon reservoir using tNavigator." *Progress in Energy and Environment* 18 (2021): 1-22. https://doi.org/10.37934/progee.18.1.122
[37] Thakkar, Ankit, and Kinjal Chaudhari. "A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions." *Expert Systems with Applications* 177 (2021): 114800. https://doi.org/10.1016/j.eswa.2021.114800
[38] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." *Neural computation* 12, no. 10 (2000): 2451-2471. https://doi.org/10.1162/089976600300015015
[39] Bae, Jinwoo, and Zhimin Xi. "Learning of physical health timestep using the LSTM network for remaining useful life estimation." *Reliability Engineering & System Safety* 226 (2022): 108717. https://doi.org/10.1016/j.ress.2022.108717
[40] Muneer, Amgad, Shakirah Mohd Taib, Sheraz Naseer, Rao Faizan Ali, and Izzatdin Abdul Aziz. "Data-driven deep learning-based attention mechanism for remaining useful life prediction: Case study application to turbofan engine analysis." *Electronics* 10, no. 20 (2021): 2453. https://doi.org/10.3390/electronics10202453
[41] Xie, Zhiyuan, Shichang Du, Jun Lv, Yafei Deng, and Shiyao Jia. "A hybrid prognostics deep learning model for remaining useful life prediction." *Electronics* 10, no. 1 (2020): 39. https://doi.org/10.3390/electronics10010039
[42] Kumar, Krishna D. "Remaining useful life prediction of aircraft engines using hybrid model based on artificial intelligence techniques." In *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1-10. IEEE, 2021. https://doi.org/10.1109/ICPHM51084.2021.9486500

[43] Zhao, Chengying, Xianzhen Huang, Yuxiong Li, and Muhammad Yousaf Iqbal. "A double-channel hybrid deep neural network based on CNN and BiLSTM for remaining useful life prediction." *Sensors* 20, no. 24 (2020): 7109. https://doi.org/10.3390/s20247109

[44] Da Costa, Paulo Roberto De Oliveira, Alp Akcay, Yingqian Zhang, and Uzay Kaymak. "Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation." *International journal of prognostics and health management* 10, no. 4 (2019).

[45] Zheng, Shuai, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta. "Long short-term memory network for remaining useful life estimation." In *2017 IEEE international conference on prognostics and health management (ICPHM)*, pp. 88-95. IEEE, 2017. https://doi.org/10.1109/ICPHM.2017.7998311