Attention-based Remaining Useful Lifetime (ARUL) prediction for Aero-propulsion Systems

Arash Noori
Carleton University
Ottawa, Canada
ashnoori@cmail.carleton.ca

Serena McDonnell
Aggregate Intellect Inc.
Toronto, Canada
serena@ai.science

Amir Amjadian
École de technologie supérieure
Montreal, Canada
amir.amjadian.1@ens.etsmtl.ca

Ehsan Amjadian
Royal Bank of Canada - RBC
Toronto, Canada
ehsan.amjadian@rbc.com

Muhammad Rizwan Abid
Florida Polytechnic University
Florida, USA
mabid@floridapoly.edu

***Abstract*—Remaining Useful Lifetime (RUL) prediction is a key component in Prognostics and Health Management (PHM) of aircraft and aeroengines. Accurate and real-time RUL prediction can increase the safety of aircraft and decrease operation and maintenance costs. Previous approaches employ variants of recurrent neural network architectures (e.g., LSTM) to encode sensor data at each time step. In other domains of artificial intelligence such as Natural Language Processing (NLP), neural attention based architectures have been shown to have superior performance compared to other available architectures in terms of capturing long-distance dependencies in time series data [17], which can play a critical role in predicting RUL. In this work, to learn RUL relationships of continually degrading equipment pertinent to an aeroengine, we propose ARUL, a neural attention based model to more efficiently capture the long-term characteristics in a sequence of sensor measurements. In the PHM domain, the development of prognostic models requires run-to-failure trajectories, which are rarely available in real safety-critical applications due to the low occurrence of failures and faults in these systems. To overcome this, the proposed method is applied to 4 datasets generated from the simulation of realistic large commercial turbofan engines. Specifically, turbofan simulation data obtained from the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPPS) is used to assess the performance of the proposed a raw Transformer architecture for estimating the RUL of turbofan engines. Additionally, we illustrated the benefits and drawbacks of different components of the Transformers architecture in the context of time-series prediction.**

***Index Terms*—PHM; RUL; CMAPSS; deep learning; transformer architecture**

I. Introduction

The monitoring and maintenance of aeroengines are of extreme importance in aerospace and aviation industries. As a consequence of poor maintenance, flight delays may occur at best and, the consequences can be fatal at worst. Additionally, the poor performance of aviation machines is considered a high contributing factor to climate change. In particular, the emission of CO_2 gets the most attention in this regard. According to [6], it's estimated that global aviation emitted 1.04 billion tonnes of CO_2 which represented 2.5% of total CO_2 emissions in 2018. This emphasizes the importance of proper maintenance and monitoring of aviation pieces of machinery, as early detection of faulty gears and repairing them will reduce fuel consumption and therefore CO_2 emissions.

Prognostic and Health Management (PHM) is an approach aimed at conducting monitoring and maintenance of aerospace systems before failures occur. A critical component of PHM involves accurately estimating the Remaining Useful Lifetime (RUL) of systems equipment. RUL refers to the amount of time left before a piece of equipment cannot perform its function as intended, and is enabled by analyzing sensor measurements place on the particular piece of equipment . Accurate RUL prognostics enable the interested parties to assess an equipment's health status and to plan future maintenance actions, such as logistics of personnel, spare parts, and services [12]. As such, RUL is essential to the PHM process. Moreover, overestimating RUL leads to unplanned failures, and sometimes catastrophic incidents, whereas underestimating RUL leads to under-utilization of the components and burden of extra cost on the industry. Traditional physics-based prognosis methods mostly employ algebraic and differential equations to model system degradation and heavily rely on the prior state of the engine and its equipment, which makes RUL prediction complex and challenging. One such example of this is the Forman crack growth law [14]. However, these approaches are not scalable, therefore, the need for prognostic models that can properly cope with the complexity, volume, and velocity of data is accentuating. Be that as it may, in the aviation industry development of data-driven models and acquiring properly labeled realistic data is very challenging due to the rarity of datasets with run-to-failure trajectories and the low occurrence of engine failures in real life. Furthermore, the sensitive nature of failures and the potential legal implications, make data acquisition from companies and manufacturers more difficult. To address these challenges, utilizing synthetic datasets that are generated with simulators in a lab environment attract great attention from academic institutions and the industry [3] to model damage propagation of a system.

Recently, deep learning (DL) has shown a great potential to process highly non-linear and varying sequential data with minimal human input within the PHM domain [22]. With the continuous developments of graphic processors and increasing volumes of data, DL methods have been exhibiting superior performance in comparison with traditional physics-based and machine learning (ML) models due to their unique and efficient capability of handling and processing high volume and complex data. Moreover, compared to statistical-based models, DL models have superior performance with their ability to better capture the complex non-linear relationship between the features (i.e., inputs and outputs). To name a few, DL algorithms based on recurrent neural networks (RNN) and long short term memory (LSTM) draw great attention in the literature [9]. These models have shown strong capabilities in apprehending short-term dependencies in time series data. Despite the fact of DL models superior performance there are a few disadvantages that need to be overcome. One of the which is the training of RNNs and LSTMs that could be challenging due to the lack of parallel computation and the issue of gradient vanishing and exploding which deprive the model of learning long-term dependencies. To address these issues, for time-series sequence modeling a self-attention-based Transformer network [17] has recently been proposed. Unlike RNNs this model does not process the data in order and enables efficient parallel computation which results in computational cost reduction and better capturing the long term-dependencies. Additionally, considering the fact that the fault in engines developed over time, holding on to the short-term dependencies before degradation occurs will increase the burden of redundant computation on the model. Moreover, the sensors signals -that indicate the health index of the engine- before degradation do not carry significant information about the status of the engine. To address this, a piecewise linear degradation model can be utilized to limit the captured and processed information of the system before the fault development points. To overcome these major limitations in existing DL models, this project will propose ARUL, a neural attention-based model to more efficiently capture the long-term characteristics in a sequence of sensor measurements.

II. Related Work

In this section, we present an overview of methods used to predict the Remaining Useful Lifetime (RUL) of an aero-engine. We focus on both physics-based and machine learning-based approaches to RUL estimation and further segment the machine learning discussion into the specific models used. We additionally note which previous works focus on our dataset of interest, C-MAPSS.

A. Physics-Based Models

A physics-based model is a representation of the governing equations of motion, most simply described by Newton as $F = ma$. These equations describe the behaviour of a physical system in mathematical terms. Such mathematical representations are used in many sub-domains, such as biology [8], geophysical fluid dynamics [10], and in this domain, RUL prediction [7].

In a seminal work, Saxena et al (2008) [16] use a physical model to describe how damage propagation can be modeled within the modules of aircraft gas turbine engines. They evaluate their performance on C-MAPSS.

B. Machine Learning-based approaches

A machine learning model is a data-driven approach where the specific parameters of a model are learned based on the underlying data. This is most simply described by the equation of a line, $y = mx + b$, where m and b are the parameters to be learned.

Recent advances have been made in machine learning, notably in the sub-domain of natural language processing (NLP). The following works focus on various NLP approaches to sequential modelling, and we segment these approaches into three categories.

1) Recurrent Neural Networks: As part of the IEEE 2008 Prognostics and Health Management conference challenge problem, Heimes [5] used a recurrent neural network (RNN) architecture to estimate RUL, and placed second in the competition. Heimes used an Extended Kalman Filter training method combined with an evolutionary algorithm for training the RNN.

Yu et al (2020) [20] use a bidirectional RNN encoder-decoder architecture, combined with curve fitting to create an ensemble method. The ensemble approach is evaluated on C-MAPSS and outperformed prior methods in term of prediction accuracy.

C. Deep Learning-based approaches

As mentioned earlier despite the great achievements of traditional methods in intelligent fault diagnosis they still fall short in analyzing complex faults when there is minimal prior knowledge, in addition, these traditional methods have poor generalization ability [19]. Deep learning models, in contrast, are able to capture deep feature representations by utilizing various non-linear transformations and non-linear functions. As such deep learning displays great potential to overcome the deficits of traditional models and have the potential to accurately predict the health state of equipment.

1) Long Short Term Memory(LSTM): Zheng et al (2017), [20] note that prior to their work, regression based approaches and convolutional neural network (CNN) approaches used features created from sliding windows to build models. They postulated that these approaches did not fully incorporate sequential information as the existing approaches could not model long-term dependencies properly. LSTM was proposed to better make sense of long-term sequential information in the data. Evaluating on C-MAPSS, the authors found an LSTM approach to outperform CNNs.

Wu et al (2018) [18] benchmarked a vanilla LSTM architecture against a vanilla RNN, and a RNN with Gated Recurrent Units (GRU). They found a significant performance increase on the C-MAPSS dataset when using an LSTM, compared to the benchmarks.

Ellefsen et al [4] explored semi-supervised learning for RUL prediction, and combined a Restricted Boltzmann machine and LSTM architecture, with a genetic algorithm used for hyper-parameter tuning. They evaluated their proposed approach on C-MAPSS and obtained improvement over older methods.

2) Attention: to the complexity of the real world system degradation mechanics, following on recent advances in NLP that focus on attention [17], de Oliveira da Costa et al (2019) [1] proposed an LSTM architecture combined with a global attention mechanism, and evaluated their proposed model on C-MAPSS. They obtain competitive results. Notably, this method, unlike other methods, does not require prior knowledge of system degradation mechanics.

Mo et al (2021) [13] combined transformer encoder's ability to capture long-term dependencies with a CNN's ability to capture local contexts at each time step. To that end, they applied a gated convolutional unit as a first layer followed by transformer encoder blocks. They used C-MAPSS and obtained superior performance compared to the previous approaches.

3) Deep Belief Networks: Zhang et al (2016) [21] propose a multi-objective deep belief network ensemble (MODBNE) method aimed at avoiding the need for handcrafted features, which was mentioned by the authors to be a drawback of previous approaches. MODBNE uses a multi-objective evolutionary algorithm applied to the traditional deep belief network (DBN) training technique to evolve multiple DBNs simultaneously, subject to accuracy and diversity as two conflicting objectives. These networks are ensembled with learnable weights, optimized via an evolutionary algorithm. They obtained competitive results on the C-MAPSS dataset.

III. Methodology

In this section, we describe the machine learning architecture used for RUL prediction. We begin by formulation the RUL prediction task, and then discuss the building blocks of the transformer encoder. The mathematical notation used in this paper is outlined in Table I.

A. RUL Prediction

The goal of RUL prediction is as follows: given a time series of sensor measurements, predict the remaining useful lifetime of that aeroengine. Mathematically stated, given n measurements $\{x_0^j, x_1^j, ..., x_n^j\}$ for j sensors, we aim to predict the remaining useful lifetime R_{n+1} at time step $n+1$, where sequential measurements i and $i+1$ are equally spaced in time.

Putting this together, the RUL prediction problem is formulated as: given n measurements $\{x_0^j, x_1^j, ..., x_n^j\}$ for j sensors attached to an aeroengine, we aim to predict the normalized remaining useful lifetime $\bar{R}_{n+1}$ at time step $n+1$. R_{n+1} as our final RUL estimate at time $n+1$ after applying an inverse transformation to remove normalization.

TABLE I
Variables used in RUL prediction.

Notation	
x_i^j	sensor j's measurement value at time step i
x_i	j-dimensional vector of measurements at time step i
R_i	remaining useful lifetime (RUL) at time step i
$\bar{R}_i$	normalized RUL at time step i

B. Transformer Encoder

We will discuss the following building blocks of the transformer encoder used for RUL prediction. Note that this a regression problem. The transformer encoder is made up of several sub-layer components. The discussed components are: linear input layer, positional encoding, the multi-head attention sub-layer, pointwise feed forward network sub-layer, and the output layer. We finally discuss the stacking of multiple transformer encoders.

1) Linear Input Layer: In [17], the first layer of a transformer encoder is an embedding layer, which learns a vector representation of each word in a sequence. In the case of time series prediction, we are not modelling a sequence words. Instead, we are modelling a sequence of sensor measurements for multiple sensors, where at each time step we have multiple measurement values.

More concretely, if we have j sensors, then at time step i in our sequence, our input is x_i which is a j-dimensional vector. In this sense, we already have vector representations at each time step, and there is no need for an embedding layer to learn representations.

To that end, the first layer in the transformer encoder is a linear layer.

2) Positional Encoding: The transformer encoder architecture does not have a notion of sequential order. In order to incorporate sequential order into the model, learnable positional encodings are added to each input vector, as in CyberBERT [11]. That is, we add p_i to each input x_i, making the transformed input to the multi-head self-attention layer (below) of the form $h_i = x_i + p_i$. This positional encoding helps the model understand what part of the input time series it is dealing with.

3) Multi-head Self-Attention: The next block in the transformer encoder is the multi-head self-attention block. The self-attention operation takes the feature vectors (the sensor measurements) as its input. For each input vector, three additional vectors are created. These are called the query, key, and value vectors. Mathematically, each feature vector is multiplied by three projection matrices $(\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V) \in \mathbb{R}^{d \times d}$.

The entries in the matrix are learned during the training process. The query, key, and value matrices for all feature vectors are represented as: $\mathbf{E}\mathbf{W}^Q, \mathbf{E}\mathbf{W}^K, \mathbf{E}\mathbf{W}^V$, where every row in the embedding matrix $\mathbf{E} \in \mathbb{R}^{(I \times d)}$ corresponds to a feature vector in the dataset (a time step of all sensor measurements). Once the three aforementioned projection matrices are created, they are each passed to a scaled dot-product attention layer to obtain attention weights. This operation is defined below:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}.$$

The term "attention head", or "head" is used to refer to a single pass of the operations described above.

In our transformer encoder, we use multi-head attention, denoted by ϕ, as the attention operation is performed multiple times for various random initialization of the query, key, and value matrices. The results of each single attention head are then concatenated, and multiplied by another weight matrix, denoted $\mathbf{W}^H$. The mathematical notation for multi-head attention is below:

$$\mathbf{S} = \phi(\mathbf{E}) = \text{Concat}(head_1, head_2, ..., head_h)\mathbf{W}^H$$
$$head_i = \text{Attention}(\mathbf{E}\mathbf{W}_i^Q, \mathbf{E}\mathbf{W}_i^K, \mathbf{E}\mathbf{W}_i^V),$$

where h is the number of heads. The output, $\mathbf{S}$, is then passed to the next sub-layer in the transformer.

4) Point-wise Feed Forward Neural Network (PFFN): The output of the multi-head self-attention sub-layer is passed to a pointwise feed forward neural network (PFFN), which adds non-linearity to the model. This additional non-linearity is beneficial as the self-attention operation is mainly based on linear projections. The PFFN is applied to the outputs of the multi-head self-attention sub-layer separately and identically at each position.

The point-wise feed forward neural network consists of two affine transformations with a Gaussian Error Linear Unit (GELU) activation applied between the transformations. GELU is used as it is smoother than the standard ReLU activation [2]. For a given input ($\mathbf{x}$), the PFFN is defined as:

$$PFFN(\mathbf{x}) = GELU(\mathbf{x}\mathbf{W}^{(1)} + \mathbf{b^{(1)}})\mathbf{W}^{(2)} + \mathbf{b^{(2)}},$$

where $\mathbf{W}^{(1)}, \mathbf{b^{(1)}}, \mathbf{W}^{(2)}$, and $\mathbf{b^{(2)}}$ are learnable parameters.

5) Output Layer: The final layer is the output layer which is a sigmoid activation function as we are performing regression.

6) Stacking: The self-attention mechanism described in the previous section is able to capture interactions between feature vectors (sensor measurements) at different time steps. Stacking transformer blocks together enables the model to learn even more complex interactions.

However, as the model becomes deeper, it becomes more difficult to train. Layer normalization and dropout are applied after each multi-head attention and PFFN sub-layer. LeakyReLU is also added after each sub-layer in order to capture the further non-linearities. We use residual connections as well between the input to the sub-layer, and the normalization layer, as in the standard transformer architecture.

Overall, the output of the multi-head attention and PFFN sub-layers is as follows:

$$\mathbf{S}' = \text{LayerNorm}(\mathbf{S} + \text{Dropout}(\phi(\mathbf{S}))),$$
$$\mathbf{H} = \text{LeakyReLU}(\mathbf{S}'\mathbf{W}^{(1)} + \mathbf{b^{(1)}})\mathbf{W^{(2)}} + \mathbf{b^{(2)}},$$
$$\mathbf{F} = \text{LayerNorm}(\mathbf{S}' + \text{Dropout}(\mathbf{H})),$$

Where $\mathbf{W}^{(1)}, \mathbf{b^{(1)}}, \mathbf{W}^{(2)}$, and $\mathbf{b^{(2)}}$ are learnable parameters, $\mathbf{S}$ is the output of the multi-head attention sub-layer, $\mathbf{S}'$ is the result of dropout and layer normalization applied to $\mathbf{S}$, $\mathbf{H}$ is the result of applying LeakyReLU after each layer, and $\mathbf{F}$ is the overall output of the self-attention and PFFN sub-layers.

IV. Experiments

A. Datasets and Pre-Processing

The lack of run-to-failure data sets is a persistent problem for data-driven prognostic models. Particularly, in the aerospace and aviation industry, real-world data typically contains fault signals for developing faults, but no or few datasets are available that capture fault progression until failure, due to the low occurrence of engine failures in real life. Obtaining real fault progression data can be time-consuming and costly. Therefore, utilizing simulated datasets that are generated with simulators in a lab environment attract great attention from academic institutions and the industry [3] to model the damage propagation of a system. With that being said, we chose to evaluate our proposed method on the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dataset that was developed at NASA [16]. The dataset [15] is publicly available and has been around for researchers to compare their approaches in the field of prognostics. The dataset comprises four distinct datasets, FD001, FD002, FD003, and FD004, which contain multivariate time series run-to-failure trajectories of a fleet of engines. Table II represents the overview of the dataset and its subsets. Each subset consists of 21 sensor measurements, the initial wear of each engine, and 3 different operation conditions. The operational settings that determine the various flight conditions of the aircraft consist of altitude, mach number, and throttle resolver angle.

TABLE II
Details of the C-MAPSS dataset.

	FD001	**FD002**	**FD003**	**FD004**
Training sequences	100	260	100	249
Testing sequences	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

Description of each sensor measurements and their corresponding units can be found in Table III. Through exploratory data analysis, we learned that not all the sensor measurements contribute to the estimation of RUL. To name a few, the total temperature at the fan inlet, the total pressure in bypass-duct, and the high-pressure turbine coolant bleed are constant values or have a high correlation with the actual labels. In both cases, the model won't benefit from their information, and as such were dropped from the dataset we used for our experiments. Additionally, to reduce the computational complexity of the model, we followed the criteria for feature selection as proposed by [13]. Specifically, the 3 operational settings, in addition to, sensor numbers {4, 8, 9, 13, 19, 21, 22, 25, 26} are excluded from the selected features. Furthermore, each subset of data is divided into two parts: training and test set. With this split, all training sequences contain complete run-to-failure trajectories, while testing sequences terminate sometime before a failure occurs. Hence, the prediction task in the test set is to estimate the remaining life cycles for each engine from the last recorded time.

We normalized the data using a MinMax scaler, as different sensor measurements might have different scales, which would lead to different numerical ranges input to our models. MinMax is a common rescaling method used in machine learning. By applying this method all the input sensor measurements and RUL values will fall within the range of [0,1].

$$\mathbf{x_i'} = \frac{\mathbf{x_i} - \mathbf{min}(\mathbf{x_i})}{\mathbf{max}(\mathbf{x_i}) - \mathbf{min}(\mathbf{x_i})}$$

Where $\mathbf{x_i'}$ denotes the original value of sensor data, and $\mathbf{x_i}$ is the normalized one.

In our datasets, RUL targets are only available at the last time step for each engine in the test datasets. However, it is reasonable to estimate the RUL as a constant value when the engines operate in normal conditions (Heimes, 2008). Similar to the works of Listou Ellefsen et al. (2019) and Lei et al. (2018), we propose a piecewise linear degradation model to define the correct RUL values in the training sets, as illustrated in Fig. 1. That is, after an initial period with constant RUL values, we assume that RUL targets decrease linearly as the number of observed cycles progresses. We denote $\mathbf{Re}$ as the constant RUL, the initial period in which the engines are still working in healthy conditions. To allow comparison to previous works, a constant $\mathbf{Re}$ of 125 cycles is selected in our experiments.

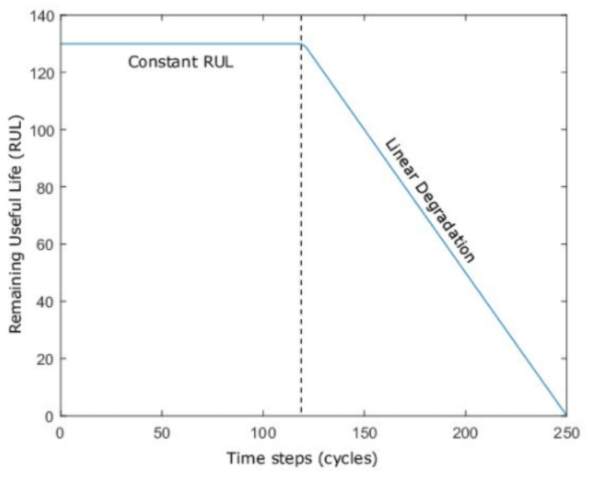

Fig. 1. piecewise Linear Degradation

1) Experimental Setup and Hyperparameter Tuning: We implemented our approach on TensorFlow GPU 2.6.0, and all the experiments are run on a personal computer with Intel Core i7-7700HQ CPU @ 2.80GHz, 16 GB RAM, and GEFORCE GTX 1060 - 6 GB gpu. The transformer encoder trained on a fixed sequence length of 21, since the largest batch of data contains 21 sequences, and the neural network weights are initialized from a random uniform distribution.

To learn the optimal hyperparameters of the model, we follow a manual Bayesian approach where baseline hyperparameter values are selected for all but one hyperparameter. For that single hyperparameter, we explore multiple values while keeping track of the effect each value has on that parameter, making sure to explore a range of small and large values. This allows us to understand the effect of each hyperparameter and its corresponding values in hyperparameter space. Lastly, through our exploration we recorded different configurations and settings to provide an ablation study for different components of transformer architecture such as attention mechanism, and the number of hidden units. Table IV illustrates highlights and their corresponding configurations.

Deep learning neural networks are prone to overfit a training dataset. One solution to reduce overfitting is to use dropout, that is, ensembles of neural networks with different model configurations and settings. This method uses a single model with various network architectures that can be achieved by randomly dropping out nodes during training, and offers an effective regularization to reduce overfitting as well as improve the model accuracy. In this work, we use dropout to regularize the network with the initial value of 0.2.

V. Evaluation

We used the Root Mean Squared Error (RMSE) to evaluate the performance of the transformer encoder. RMSE is chosen as it is used commonly in the RUL prediction literature. RMSE is defined as:

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

Where y_i is the ground truth RUL and $\hat{y}_i$ is the predicted RUL, and N is the number of samples.

VI. Results and Findings

In this section, we report the results and findings of our experiments. The performance of the model is being evaluated by RMSE (i.e., a higher RMSE indicates a lower model performance). We observed that by increasing the number of hidden units, the accuracy of training, validation, and test increased considerably. Initially, the RMSE on the FD001 test dataset with 128 units was 32.77, and by increasing the units to 256, and 512, the RMSE improved accordingly to 29.96, and **15.37**, respectively. This as expected implies that by increasing the number of neurons in a deep learning network, the model capacity increases, thus more complex functions can be learned by the network, which in this case has resulted in higher model performance. However, this is not necessarily always true. By further tweaking the model, we observed that increasing the number of attention heads does not carry the same results as the number of hidden units. The results indicate that by adding 1 transformer block to the stack of 2, the recorded RMSE significantly

TABLE III
DESCRIPTION OF SENSOR MEASUREMENTS.

Sensor number	**Description**	**Units**
1	Operational settings	–
2	Operational settings	–
3	Operational settings	–
4	Total temperature at fan inlet	°R
5	Total temperature at low pressure compressor outlet	°R
6	Total temperature at high pressure compressor outlet	°R
7	Total temperature at low pressure turbine outlet	°R
8	Total pressure in bypass-duct	psia
9	Total pressure at high pressure compressor outlet	psia
10	Pressure at fan inlet	psia
11	Physical fan speed	rpm
12	Physical core speed	rpm
13	Engine pressure ratio	–
14	Static pressure at high pressure compressor outlet(Ps30)	psia
15	Ratio of fuel flow to Ps30	pps/psi
16	Corrected fan speed	rpm
17	Corrected core speed	rpm
18	Bypass ratio	–
19	Burner fuel–air ratio	–
20	Bleed enthalpy	–
21	Demanded fan speed	rpm
22	Demanded corrected fan speed	rpm
23	High pressure turbine coolant bleed	lbm/s
24	Low pressure turbine coolant bleed	lbm/s

TABLE IV
ABLATION STUDY AND CONFIGURATIONS.

Embedding dimension	128	256	256	256	**512**	512	512	512	512
Number of attention-heads	1	1	2	2	**2**	2	4	4	2
Feed-forward dimension	128	256	256	256	**512**	512	512	512	512
Dropout rate	0.3	0.3	0.3	0.3	**0.3**	0.3	0.3	0.3	0.3
Activation function	relu	relu	relu	sigmoid	**sigmoid**	sigmoid	sigmoid	sigmoid	sigmoid
Number of transformer-block	1	1	1	2	**2**	2	2	3	4
Batch size	128	128	256	256	**256**	512	512	512	256
Number of epochs	100	100	100	100	**100**	100	100	100	100
RMSE	32.77	30.68	29.96	23.47	**15.37**	15.6	17.41	25.74	17.63

Note: the reported RMSE is on FD001.

increased from 19.19 to 25.74. Additionally, by increasing the number of attention heads from 2 to 4, the RMSE rose considerably from 15.6 to 17.41. This implies that as the backpropagation algorithm calculates derivatives using the chain rule, although increasing the depth of the network is expected to improve the performance of the model, the higher number of parameters often leads to the vanishing gradient. More sample size definitely could help to improve the performance and overcome this challenge more appropriately. Other observation is that the Transformer model performs better when sigmoid activation function is used instead of relu. With that being said, the best results by the proposed algorithm were obtained with 512 hidden units, 2 transformers blocks, 2 attention heads, and the RMSE of 15.3, that is the highlighted configuration in Table IV. More details on the optimal number of hyperparameters also can be found in Table IV.

VII. CONCLUSION

In this work, we proposed a Transformer architecture for RUL prediction of turbofan degradation engines using the C-MAPPS datasets. We proposed an architecture containing an attention mechanism that has been used to visualize the temporal relationships between inputs and predicted RUL outputs. As the scope of this study was to provide various ablation studies to display the benefits and drawbacks of the Transformer architecture components, the results show how changing the depth of the network, or how hyperparameter tuning will affect the overall performance of the model. However, to run a fair comparison with previous works, it is beneficial to reproduce the reported state-of-the-art results of the earlier studies and compare the proposed method in this paper against them, thus this is a work in progress. Lastly, it is notable that there are higher performance models reported recently, however, quite a few of their codes are not available and not reproducible due lack of maintenance. Therefore, we provided the proposed method code of the raw Transformer architecture for time-series prediction available in Jupyter notebook version on GitHub[1] for educational purposes, further experiment, and future research reproducibility.

[1]https://github.com/Ashnoori/ARUL

References

[1] Paulo Roberto de Oliveira da Costa et al. "Attention and long short-term memory network for remaining useful lifetime predictions of turbofan engine degradation". In: *International Journal of Prognostics and Health Management* 10.4 (2019).

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Ömer Eker, Fatih Camci, and I.K. Jennions. "Major Challenges in Prognostics: Study on Benchmarking Prognostics Datasets". In: July 2012.

[4] André Listou Ellefsen et al. "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture". In: *Reliability Engineering & System Safety* 183 (2019), pp. 240–251.

[5] Felix O Heimes. "Recurrent neural networks for remaining useful life estimation". In: *2008 international conference on prognostics and health management*. IEEE. 2008, pp. 1–6.

[6] Corinne Le Quéré et al. *Global Carbon Budget 2018*. Dec. 2018. URL: https://essd.copernicus.org/articles/10/2141/2018/essd-10-2141-2018.html.

[7] Yaguo Lei et al. "Machinery health prognostics: A systematic review from data acquisition to RUL prediction". In: *Mechanical systems and signal processing* 104 (2018), pp. 799–834.

[8] Alfred J Lotka. "Contribution to the theory of periodic reactions". In: *The Journal of Physical Chemistry* 14.3 (2002), pp. 271–274.

[9] Arnaz Malhi, Ruqiang Yan, and Robert Gao. "Prognosis of Defect Propagation Based on Recurrent Neural Networks". In: *IEEE T. Instrumentation and Measurement* 60 (Mar. 2011), pp. 703–711. DOI: 10.1109/TIM.2010.2078296.

[10] Serena McDonnell. "Circulation and dynamics over the shelf-slope in response to idealized forcing". MA thesis. Hong Kong University of Science and Technology, 2017.

[11] Serena McDonnell et al. "CyberBERT: A Deep Dynamic-State Session-Based Recommender System for Cyber Threat Recognition". In: *2021 IEEE Aerospace Conference (50100)*. IEEE. 2021, pp. 1–12.

[12] George Michalos et al. "Automotive assembly technologies review: Challenges and outlook for a flexible and adaptive approach". In: *Cirp Journal of Manufacturing Science and Technology* 2 (Dec. 2010), pp. 81–91. DOI: 10.1016/j.cirpj.2009.12.001.

[13] Yu Mo et al. "Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit". In: *Journal of Intelligent Manufacturing* (2021), pp. 1–10.

[14] Charles Oppenheimer and Kenneth Loparo. "Physically based diagnosis and prognosis of cracked rotor shafts". In: *Proceedings of SPIE - The International Society for Optical Engineering* (July 2002).

[15] *Prognostics Center of Excellence - Data Repository*. 2013. URL: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[16] Abhinav Saxena et al. "Damage propagation modeling for aircraft engine run-to-failure simulation". In: *2008 international conference on prognostics and health management*. IEEE. 2008, pp. 1–9.

[17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[18] Yuting Wu et al. "Remaining useful life estimation of engineered systems using vanilla LSTM neural networks". In: *Neurocomputing* 275 (2018), pp. 167–179.

[19] Gaowei Xu et al. "Data-driven fault diagnostics and prognostics for predictive maintenance: A brief overview". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 103–108.

[20] Wennian Yu, II Yong Kim, and Chris Mechefske. "An improved similarity-based prognostic algorithm for RUL estimation using an RNN autoencoder scheme". In: *Reliability Engineering & System Safety* 199 (2020), p. 106926.

[21] Chong Zhang et al. "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics". In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2306–2318.

[22] Guangquan Zhao et al. "Research advances in fault diagnosis and prognostic based on deep learning". In: *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*. 2016, pp. 1–6. DOI: 10.1109/PHM.2016.7819786.