AI

MDPI

Article

Remaining Useful Life Prediction Using Temporal Convolution with Attention

Wei Ming Tan [1,†] and T. Hui Teo [1,2,*,†]

[1] Engineering Product Development Pillar, Singapore University of Technology & Design, 8 Somapah Road, Singapore 487372, Singapore; dylan_tan@mymail.sutd.edu.sg
[2] Science, Math, & Technology Cluster, Singapore University of Technology & Design, 8 Somapah Road, Singapore 487372, Singapore
* Correspondence: tthui@sutd.edu.sg
† These authors contributed equally to this work.

Abstract: Prognostic techniques attempt to predict the Remaining Useful Life (RUL) of a subsystem or a component. Such techniques often use sensor data which are periodically measured and recorded into a time series data set. Such multivariate data sets form complex and non-linear inter-dependencies through recorded time steps and between sensors. Many current existing algorithms for prognostic purposes starts to explore Deep Neural Network (DNN) and its effectiveness in the field. Although Deep Learning (DL) techniques outperform the traditional prognostic algorithms, the networks are generally complex to deploy or train. This paper proposes a Multi-variable Time Series (MTS) focused approach to prognostics that implements a lightweight Convolutional Neural Network (CNN) with attention mechanism. The convolution filters work to extract the abstract temporal patterns from the multiple time series, while the attention mechanisms review the information across the time axis and select the relevant information. The results suggest that the proposed method not only produces a superior accuracy of RUL estimation but it also trains many folds faster than the reported works. The superiority of deploying the network is also demonstrated on a lightweight hardware platform by not just being much compact, but also more efficient for the resource restricted environment.

Keywords: attention; deep learning; convolution neural network; multivariate time series; prognostics; remaining useful life

Citation: Tan, W.M.; Teo, T.H. Remaining Useful Life Prediction Using Temporal Convolution with Attention. *AI* **2021**, *2*, 48–70. https://doi.org/10.3390/ai2010005

Received: 13 December 2020
Accepted: 31 January 2021
Published: 14 February 2021

1. Introduction

The field of prognostics is centered around attempting to accurately predict the amount of time left before an equipment of component fouls. The RUL of a subsystem or component is commonly estimated for prognostics and prediction. Such methods of estimation usually make use of a series of sensors constantly monitoring machine conditions for faults. These sensor readings are very important to engineering maintenance procedures in many different industries such as aerospace, manufacturing and automotive and they initially serve as the foundation of corrective or preventive maintenance [1]. Prognostics and RUL prediction on the other hand may be key to maximizing operational and system reliability. If trustworthy RUL is obtained, a strong and fixed maintenance strategy may be developed around it that would ultimately reduce maintenance costs by offering industries that rely heavily on their equipment optimized operating efficiency, reduced sudden downtime and hence maximum cost savings.

Generally prognostics techniques requires a more hand engineered [2] modeling approach or utilizing a data driven learning method such as Artificial Neural Network (ANN) [3] where labelled data might be required. A strong know how and deep knowledge of the system is required to ultimately make a reliable model in hand engineered approaches. Machine Learning (ML) methods on the other hand, due to its bank of data,

allows such models to reliably learn inter dependencies much more quickly to acceptable degrees of accuracy. ANN in particular have shown that it is capable of learning highly non linear and complicated relations from multidimensional inputs without requiring strong prior knowledge to the system or its physical activity. Unfiltered and unprocessed inputs might even be able to produce high levels of accuracy.

In DNNs, forecasting is no new phenomena. The model predicts a future time series based on their historical data. Some examples of this exists in stock forecasting, generative audio networks [4], clinical time series analysis from recorded databases [5] or even influenza epidemic prevalence prediction [6] using past observed data such as climate or virological surveillance. Prognostics and health management problems such as predicting the RUL using the sensor monitoring data can also be modelled as a forecasting problem of a MTS.

When it comes to MTS, Recurrent Neural Network (RNN) are often used. Amongst other types of ANN, RNN provide a good level of effectiveness and robustness to sequential and even multi series models. In prognostics, RNNs and ANN are not new to the game. However, while RNNs provide promising results in prognostics applications [3], RNNs tend to be large and heavy consisting of many parameters to train on top of long gradient paths that result in lengthy training times. RNNs intrinsically sequential computation prevents effective parallelization across the time axis of the input data while potentially creating a vanishing gradient problem [7]. Recurrent networks are no stranger to forecasting and they also exists in general applications such as trajectory prediction for cars where [8] used robust recurrent networks called Long Short-Term Memory (LSTM)s to complete the estimated movement of a car on top of object detection. An LSTM was also applied to pick out anomaly in three-shaft gas turbines using skewed data that had rare occurrence of fault data [9]. The LSTM network had shown to outperform traditional classifier methods such as Support Vector Machine (SVM).

Attention mechanisms [10] are getting a lot of attention as provide promising results. The use of the self-attention [11] in seq2seq encoder decoder based transformer models [12] not only shows a great success in Natural Language Processing (NLP) applications, but becomes significantly more appealing than RNNs in many aspects. The attention mechanisms also find its way into MTS applications such as in [5,6] and even hybrid applications with RNNs [13,14]. In this paper, a temporal convolutions and a non-standard attention mechanism are proposed across the output of the stacked convolutions.

Related Work

As the problem of prognostics can be extremely variational in terms of the environment and conditions, a data set can be vastly differing. This leads to a large field of possible solutions for different applications. One of the earlier methods of applying ANN to RUL prediction and prognostics, reference [15] applies Multi Layer Perceptron (MLP) to estimate the ball bearings RUL. Reference [16] used ANN to estimate RUL of condition monitoring equipment. Some earlier studies also made use of RNNs to predict RUL [3]. A Kalman filter was used on top of an RNN to attain satisfactory results. CNN implementations also produce promising results [17]. An Extreme Learning Machine (ELM) was also used to predict the failure of the machinery without assuming homogeneous pattern [18].

As ML methods attempt to capture highly complex and multi dimensional inputs, some attempted to use multiobjective evolutionary algorithms together with Deep Belief Network (DBN)s to estimate RUL of a system. Multiobjective Deep Belief Networks Ensemble (MODBNE) was employed by [19] that achieved good results. However as longer dependencies comes into the play, other types of networks are required. LSTM units were used to learn temporal data and predict RUL [20]. A vanilla LSTM performed well against standard RNNs and simpler alternatives of the LSTM on the same C-MAPSS data set that are used for this study. More research on using LSTM for the prognostics were also done in [21]. This shows that LSTM is more capable in estimating RULs for complex engineering systems. References [22,23] also used a bi-directional LSTM on a

health index computation of the raw data, showing significant improvements to RUL prediction accuracy. An improved CNN model was also proposed in [24] that uses an even deeper convolution model than in [17]. This method contained more convolutions and filters per layer, thus retaining more information per depth, giving more focus on the finer details which as a result performed exceptionally well. The more successful models [22,24] were not as shallow, thus suffers from high computational load which meant longer training times and slower deployment. Reference [25], like us, proposed temporal convolutions, however on top of recurrent LSTM cells. Other neural network methods may include using a sparse auto-encoder with logistic regression to predict the RUL [26] while [27] used a Directed Acyclic Graph (DAG) containing parallel LSTM and CNN architectures to solve their prediction problem. This former displayed satisfactory performance, however it uses multiple steps and optimisations on top of a neural network. The latter however has shown very promising performance. Exceeding the deep neural network proposed in [24]. A Deep Survival Model (DSM) based on regression and discrete Weibull distribution was also used on top of a combined LSTM and CNN model [28]. This method also uses temporal convolutions and resulted in an encouraging results on all the data subsets. It also provides an alternative approach to just neural networks. Semi-supervised methods [29] have also been used to tune the starting weights which had shown further improvements in performance which shows viable methods outside of supervised neural network training. Finally a deep convolution Generative Adversarial network was integrated with an auto encoder that produced outstanding results on the same data sets [30].

Other works on time series data sets makes use of different architectures of CNNs, RNNs and attention. Chiller fault data sets have shown that one dimensional temporal convolutions have achieved good results [31]. While using two layers, LSTMs have also outperformed other forms of RNNs in detecting faulty conditions in chillers as well. Some literature specifically use a hybrid architecture of LSTM, CNN and attention to get very promising results in MTS data sets [13,14]. While they differ in some areas, reference [14] specifically demonstrates through its detailed examples how each part of the architecture is key to the success of the network. Temporal Convolution Network (TCN) are also seen to be used on time series data sets. TCNs consists primarily of one dimensional causal and dilated convolutions and perform comparably to bulky RNNs whilst remaining largely simple [32]. These literatures provide alternative approaches to MTS data sets and offer not only improved performance but also lighter networks.

2. Materials

In this section, the data set and the relevant methods that are used to arrive at the results are discussed in detailed. The C-MAPSS is an engine performance degradation data set that consists of simulated turbofan engines run to failure. It consists of four main data subsets and its details will be presented before going into details of the proposed architecture. The proposed architecture if not explicitly stated is also referred to as 'CNN + ATT' model. More information about the model will also be presented in the later section Methods. The performance of the model will be bench marked against other neural network models in the related work described above within all data subsets. The effectiveness of the different parts of the model will also be investigated to understand the strengths of the proposed method. Experimental results are taken from a mean of 5 tries. All experiments are trained and deployed on a single core hyper threaded Processor @2.3 GHz, 13 GB RAM and a Tesla K80 GPU is used. Some results will be carried out on a lightweight System on Chip (SoC) hardware platform, Raspberry Pi 3B to assess the models capability on such devices [33].

2.1. Data Set

The data set used for the proposed method is the NASA's C-MAPSS for engine performance degradation tracking and RUL prediction. The data set is generated from a C-MAPSS commercial turbofan engine simulator [34,35].

NASA C-MAPSS

The C-MAPSS data set is divided into four subsets shown in Table 1. Each subset is again further divided into training and test sets of multiple MTS. The data set is generated from an aircraft gas turbine engine and each of the subsets use a different engine that each start with its own degrees of initial wear and manufacturing variation. These variations are not made known to us. What is established however is that each data set is measured under known operating conditions as shown in Table 1. Data set 001 and 003 both have 1 operating condition while 002 and 004 have 6 conditions that makes them much more complex which also explains its much larger training sample size as compared to the sets with only 1 operating condition. The fault conditions of the data sets also vary from 1 to 2, having an increased complexity with more faults introduced. The maximum and minimum cycle sequence length for each data subsets' test and train sets are also detailed for reference.

Table 1. Subsets of C-MAPSS data set.

Data Set	**NASA C-MAPSS**			
	FD001	***FD002***	***FD003***	***FD004***
Train sets	100	260	100	249
Test sets	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2
Train Samples	17,731	48,819	21,820	57,522
Min/Max cycles for Train set	128/362	128/378	145/525	128/543
Min/Max cycles for Test set	31/303	21/367	38/475	19/486

The data sets includes 26 columns that include engine ID, time cycle, three operational sensor settings, and 21 on-board sensor measurements. The sensors are used to monitor the engine performance through measurements of speed, temperature, and pressure at different locations [34]. All sensors also possess non trivial noise, characteristic to its own measurement technique, manufacturing assembly variation and health.

All engines operate in normal condition at the beginning of the simulations. Fouling and corrosion is introduced at some point and hence start to degrade the engine randomly in the time series. The degradation will continue in the training sets and will end each cycle of an engine with complete failure. The degradation in the test sets however may not end in failure so that the robustness and accuracy of the model can be tested. The true RUL of the test sets are given for the final time step of each engine cycle. Hence the goal of the model is to estimated the RUL of the test set as close to the given true values as possible. More information about the turbofan engine data set may be found in Appendix B.

2.2. RUL Estimation

The training data subsets do not come with a given RUL. To tackle this problem and estimate the RUL of the test set from a supervised learning approach, the RUL of the train set must be modelled well. Prognostics algorithms can essentially be modelled as regression problems. However, in practical prognostics setting, this problem is far from regular regression. The RUL of the training set is not known and an inherent obstacle in prognostics problems itself is to estimate the RUL of each data point.

It is very challenging to accurately model the health of a component without a well transcribed physics model of the entire system. While a method would be to simply assign the desired output as the actual time left before functional failure [36]. This however

suggests that the health of a system degrades linearly with time which can be unrealistic and detrimental to learning accuracy of a trained model. A model would hence not be able to perform well on a test set if its training data RUL is badly represented. An alternative would be to estimate the RUL based on a suitable model. Reference [3] had shown using a MLP that it might be acceptable to use a constant RUL value in normal operation.

A Piece-wise linear degradation model with a constant R_c value was proposed, seen in Figure 1. This meant that as [3,37] had suggested and tested, the estimated value of RUL for the training set remains constant at an upper limit of $R_c < 125$. Many related literature that have achieved good performance have used this approach and the range of R_c commonly span from 120 to 135 [3,17,20,22,24,27,30]. Offering quick and reliable test performance, it would seem estimating the RUL of the train set with this model will be favourable. A limitation due to the training set's implemented piece-wise constant RUL estimation is that the model's output predictions would not estimate larger than the constant R_c. To normalize this deficiency in the training model, test RULs are rectified to match the maximum values so that a fair analysis of the model may be made in the experiments. This estimation of RUL is crucial to success of the network.

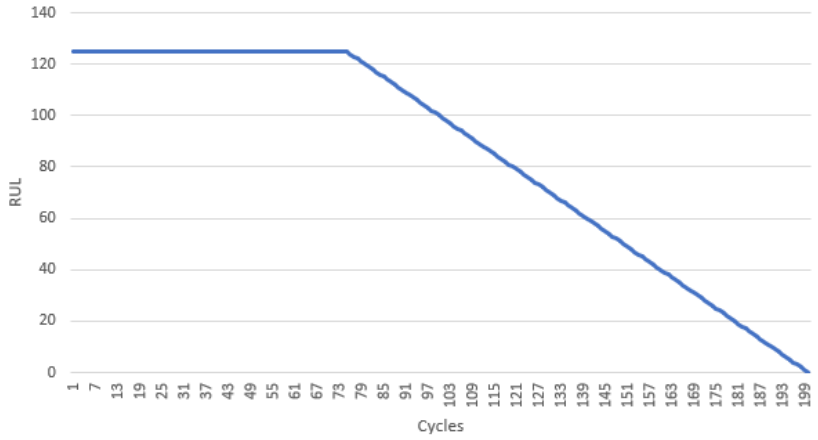

Figure 1. Piece-wise linear RUL estimation of the train set.

2.3. Metric

A couple of metrics were used in the experiments. Of which, an asymmetric scoring function was used to measure model success. This function is common to many other literature and was also suggested by [34]. This scoring function s_i for error is shown in Equation (1) and the total sum s in Equation (2). Where $d_i = \hat{y}_i - y_i$ ($RUL_{Predicted} - RUL_{True}$)

$$Score = S_i = \begin{cases} e^{-\frac{d_i}{13}} - 1, & \text{for } d_i < 0. \\ e^{\frac{d_i}{10}} - 1, & \text{otherwise.} \end{cases} \tag{1}$$

$$s = \sum_{i=1}^{N} s_i \tag{2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |d_i| \tag{4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} d_i^2}{\sum_{i=1}^{n} (y_i - y_{mean})^2} \tag{5}$$

This function penalizes late predictions more than early ones. In practical prognostics applications, a late results would mean that maintenance procedure is scheduled late. This risk in the industry could lead to very costly results. Early predictions however pose less problems as it would mean that the machine had not failed yet. Overall, one would attempt to minimize this score as a large number would indicate low accuracy. This would be a significant equation in the prognostics problem and most literature will present its metric as a measure of model validation. Two more functions, Root Mean Square Error (RMSE) in Equation (3) and the Mean Absolute Error (MAE) at Equation (4) are used to measure accuracy of the resulting RUL. The use of these two functions will give us a better insight into the model's performance. Having a root of the average squared errors will have different outcomes for RMSE as to MAE. Since the errors are squared before they are averaged, the RMSE gives a stronger emphasis to larger errors. Hence the RMSE should be more useful when large errors are particularly undesirable, which is consistent to prognostics. This is definitely undesirable when predicting RUL. The MAE however is a more universal function and gives equal penalty to both early and late predictions, it might still provide valuable information to include this. The R-square correlation (R^2) Equation (5) is also used to benchmark model performance. This allows us to measure the strength of the relationship of the model estimations to the true values.

2.4. Proposed Architecture

The proposed DL architecture for estimating RUL on the data set will be presented in the following subsection. On a broad level, the architecture consists of four stacks of 1 dimensional convolutions across the time axis only. At the end, the features are fed into a non standard attention mechanism that would later output the RUL prediction. The preliminaries of main components that consist of the CNN and the attention mechanism are discussed before going deep into the architecture itself.

2.4.1. Convolution Neural Network

CNNs have displayed a strong ability to learn salient patterns in input data. Deeper layers are able to pick out the local salience of the signals and earlier layers are able to learn higher level representations of the inputs. CNNs were first proposed by Lecun for image processing and had showed good promise [38]. It was largely re-popularized by Krizhevsky's AlexNet many years later in its overwhelming success in the ImageNet challenge.

The two main components of CNNs are the shared weights and the pooling. The convolution filters run through the input data to extract and learn local features that are significant to the output. It is because it is able to learn spatial relations so well that, two dimensional convolutions do so well with images. CNNs have however also achieved great success in other areas such as language processing, sentence classification [39] or even larger CNN architectures used for time series classification [40].

Convolution in the time domain is not so different as well. Temporal convolutions often refer to filters run across the time axis. The TCN is a deliberately kept simple architecture that combines some of the more successful practices in recent research [32]. Reference [17] also used convolution in a similar manner on the C-MAPSS data set that is used for the experiments as well and outperformed many older algorithms and even a MLP.

Temporal convolutions were also used in [24] on the C-MAPSS data set and achieved great success. They however used the convolution in a fundamentally different way. While seemingly using two dimensional convolutions across the input data, the filters were in fact one dimensional down each time series. This means is that each filter only convolves down one time series and does not overlap with the others. Each filter generates a two dimensional output thus ultimately stacking all filters into a three dimensional output per convolution. This is a rather interesting method that had outperformed even the related works that used LSTM.

2.4.2. Proposed Convolution

The convolution used in this paper are one dimensional temporal convolutions in the time axis. Given an input MTS sequence $X = \{x_1, x_2, \ldots, x_w\}$, where $x_i \in \mathbf{R^s}$ represents the observed set of multi variable values at time i and x_w is the last input in the sequence length. Should there be an s number of input sensor sequences, total input would be $X \in \mathbf{R^{w \times s}}$. The one dimensional convolutions will have k number of filters, each $C \in \mathbf{R^{1 \times w}}$. The filter only convolves over one time series at each instance. Hence it gets shielded from noise between variables during the same time step and it also learns temporal information within the sequence length that is extremely valuable to prognostics applications. This part, while sounding trivial plays a key role in the network success as related works with temporal convolutions often convolve across the MTS data.

2.4.3. Attention Mechanism

The attention mechanism enables a model to focus in on important pieces of the select feature space. It works off paying greater attention to subsets the data to get more optimal results. Attention has shown success in NLP, image processing and even MTS applications. The general mechanisms of attention may be roughly summed up into 3 parts.

- Calculating alignment scores;
- Calculating weighted average attention vector;
- Calculating the context vector.

In earlier mechanisms, Luong had proposed three methods of calculating the scoring function [10]. The functions are the dot, general and the concatenation alignment scoring. The scoring function, Equation (7), also used in [41,42] for document word and sentence classification is rather similar to both Luong's concat and Bahdanau's proposed one in [10,43]. h_i is the input vector at the attention input. As the scoring function (7), u_i is essentially a MLP to extract hidden representations of h_i. The scoring function is multiplied by trainable v_a to measure its similarity. v_a can be viewed as a high level representation of a fixed query. Initialized and learned together with the rest of the model. Ultimately, the attention weight vector α_i are normalized with a softmax of the scores (6). The context vector is the weighted sum of input vector of each time step based on the attention weights (9).

$$\alpha_i = \frac{\exp\left(u_i \cdot v_a\right)}{\sum_t \exp\left(u_i \cdot v_a\right)} \tag{6}$$

2.4.4. Proposed Attention

The proposed attention makes some alterations to how the attention weights are calculated. Unlike the traditional attention mechanisms, the sigmoid activation function (8) is used instead of the softmax. As the softmax function normalizes the weights, it diminishes the probability that more than one variable could be useful for prediction which might often be the case in a multivariate time series. This step allows the attention mechanism in the model to pick out the features better.

$$u_i = tanh(w_a h_i + b_w) \tag{7}$$

$$\alpha_i = sigmoid(v_a \cdot u_i) \tag{8}$$

$$v_j = \sum_{i=1}^{i=k} h_i \cdot \alpha_i \tag{9}$$

2.4.5. Proposed Network Structure

The structure of ANN consists of the proposed convolution layers followed by the proposed attention. The general summary of the network can be seen in Figure 2. Given

input data of sequence length w with s number of sensor variables, $X \in \mathbf{R^{w \times s}}$. The one dimensional temporal convolution layer has k filters of size $C_i \in \mathbf{R^{1 \times w}}$. Each filter only strides 1 which means that the filters will only learn about each individual time series at a time. The convolution ends with a *tanh* activation after every layer and will generate an output $H^c \in \mathbf{R^{s \times k}}$ where c corresponds to the depth of convolution layer.

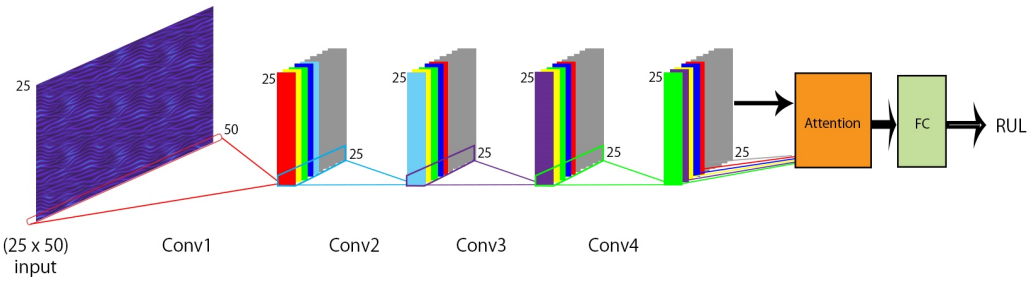

Figure 2. Proposed architecture. This architecture consists of a multivariate time series input into 4 convolution layers with tanh activation. The attention layer follows the output of the final convolution with a dropout. Finally one fully connected layer leads to an end node with linear activation for the RUL prediction.

The attention layer will attend to the columns h_i of H^4 output where $H^c = \{h_1, h_2, \ldots, h_k\}$. h_i will be multiplied with weight matrix w_a and bias b_w. After which it will be multiplied and summed with weight vector v_a to get attention weighted probabilities for each kernel i in k. Finally the attention weight α_i is weighted summed with input rows to generate output per sensor variable v_j where $j \in [1, s]$. The output context vector $V \in \mathbf{R^{s \times 1}}$ goes into a fully connected layer and after that 1 node with a linear activation to estimate RUL.

The model is made up of four stacks of the proposed convolution with an altered attention mechanism at the end. The permutation of the network structure would lead to its performance on the data subsets in the next section and its significance would be reiterated during investigations of the network.

3. Results and Discussion

In this section, the prognostics performance of the proposed architecture in estimating RUL on the C-MAPSS data set is presented. The proposed architecture which consists of one dimensional temporal convolutions with attention mechanism is used to predict the RUL values of the unseen test sets of all the four data subsets. In similar practice, the prediction methods validated on the C-MAPSS data set from the related works can be found in Table 2. The best three performing models for each sections are highlighted with bold text.

As summarized in Table 2, one would notice the general trend of improving metric towards the more recent few architectures that were in the literature toward the bottom of the table. The proposed convolution with attention network architecture displays encouraging results. While not the top in all subsets, it does do best in some and would appear to be performing better in all subsets. The proposed model is also much lighter than some of the related works. Performance generally decreases on the complex six operating conditioned data subsets 002 and 004 with 003 and 004 generally fairing slightly worse than its paired counterpart due to its additional fault condition. In order to create a fair comparison with other methods in the literature, similar metrics were used. A lower RMSE would indicate a higher RUL prediction accuracy as it implies a smaller average difference between actual and predicted RUL. The score is the total sum of errors following the asymmetrical score function mentioned in metric section Equation (1). It penalizes late predictions more than early ones so that a good model would prefer to predict early such that maintenance may be enacted, at worst, earlier to avoid catastrophic failures. Using these methods, the proposed model is benchmarked with these results from the related works.

Table 2. Performance comparison with related work.

Method	FD001		FD002		FD003		FD004	
	RMSE	*Score*	*RMSE*	*Score*	*RMSE*	*Score*	*RMSE*	*Score*
MLP [17]	37.56	18,000	80.03	7,800,000	37.39	17,400	77.37	5,620,000
SVR [17]	20.96	1380	42.0	590,000	21.05	1600	45.35	371,000
RVR [17]	23.80	1500	31.30	17,400	22.37	1430	34.34	26,500
CNN [17]	18.45	1290	30.29	13,600	19.82	1600	29.16	7890
ELM [18]	17.27	523	37.28	498,000	18.47	574	30.96	121,000
LSTM [21]	16.14	338	24.49	4450	16.18	852	28.17	5550
DBN [19]	15.21	418	27.12	9030	14.71	442	29.88	7950
MODBNE [19]	15.04	334	25.05	5590	12.51	422	28.66	6560
RNN [24]	13.44	339	24.03	14,300	13.36	347	24.02	14,300
DCNN [24]	12.61	274	22.36	10,400	12.64	284	23.31	12,500
BiLSTM [23]	13.65	295	23.18	4130	13.74	317	24.86	5430
DAG [27]	11.96	229	20.34	2730	12.46	535	22.43	3370
DSM (Regression) [28]	14.04	310	15.15	1080	14.62	325	21.92	2260
Semi-Supervised [29]	12.56	231	22.73	3366	12.10	251	22.66	**2840**
DCGAN + AE [30]	10.71	174	19.49	2982	11.48	273	19.71	3874
Proposed Model (CNN+ATT)	11.48	198	17.25	1144	12.31	251	20.58	2072

3.1. Analysis

The different intricacies of the parameters in the proposed model (CNN + ATT) are also discussed in this section.

3.1.1. Optimal Model

While deploying the proposed network, two chosen parameters seemed to play a role in performance of the network. They are the RUL constant R_c and the input sequence length w. It is found that RUL constant R_c of 125 gives the best performance for the model. This value, which is adopted by some of the related works was not always the value of choice for other. 125 was used in training the proposed model and provided the performance seen in Table 2. These same results use a w that correspond to each specific test data subset and the minimum sequence length that may be found in Table 1. The w values for the benchmarked results obtained above, w_b, are 30, 20, 30 and 15 respective to each of the four data subsets. These values of w would include all test points and is necessary for fair bench marking against other literature. However investigations with w had shown that longer sequences seemed to provide better predictions [28]. Reference [27] had also explored these differences and had come to similar conclusions. A w length of 50 had been found to be optimal for the proposed model where longer sequences had no significant improvements. As such investigations with these parameters are conducted and three networks from the literature were recreated and adapted with the same parameters to benchmark together with the experiments for better understanding of network performance.

These models are chosen due to its similar nature to the network and ease of re-training and deploying. They are trained via supervised learning and deploys neural network ML unlike more traditional methods such as gradient boosting or support vector machines. These related works also pre-processed the input training data with the piece-wise linear model.

1. Adapted CNN [17]
   A simple two layer convolution network that is adopted from [17]. This network, similar to the proposed model, uses one dimensional convolutions. They also use pooling layers that drastically reduce the data after every layer. It is trained with stochastic gradient descent. It should be noted that the convolutions for this model span across each time step and crosses multiple time steps at once, thus learning relations between close proximity time steps. This model is chosen as it out performed many of the traditional methods like Support Vector Regressor (SVR) and the MLP. It

also shared some similarity to the proposed model that would make it a more relevant comparison. Reference [17] achieves its performance using a sequence length of 15 and a piece-wise cut off limit of 130. The deployment of the network attempts to follow the literature as closely as possible, using same number of layers and convolution parameters.

2. LSTM
   The LSTM RNN model used to benchmark is a vanilla 2 stack LSTM. Each LSTM layer has a 0.2 dropout rate and hidden nodes of 100 and 50 for each respective layer. LSTMs are a rather reliable variant of the traditional RNN. They are more capable of learning long term dependencies and suffer less from the well known issues in RNN like vanishing gradients. Not only have LSTMs proven themselves to possess a strong aptitude in learning temporal patterns, it is for this very reason that they are often compared to models with attention applied. There are also many literature that uses some form of the LSTM. Hence, a simplified version of it was deployed, the size of the model was specifically chosen to be more similar in size to the Deep Convolutional Neural Network (DCNN) model next, however they tend to be very large due to its recursive nature.
3. Adapted DCNN [24]
   This model adapted from [24] was one of the first to achieve very good performance on the data set, close to the recent works metrics. It performs a two dimensional convolution on the data set. There are five layers of convolution, each of 10 filters producing a three dimensional output every time. As a result, it extracts more detail and as such learns better than its predecessors. The literature uses varying w for each subset of data and a fixed R_c of 125. The model was deployed with only 14 sensors that were deemed useful. This model shows reliable success on this prognostics data set and would be a good benchmark to the proposed model. This was recreated as closely as possible to the literature aside from choice of some parameters such as R_c.

These works are trained deployed as closely as possible with R_c fixed at 125 and w at 50. The prognostic tests are then carried out with the all the models and its comparative results recorded and discussed below. The benchmark against other networks are collated in two tables below. The models are 10 fold cross validated on the training subsets. The resulting RMSE and MAE along with their standard deviations are recorded in Table 3. Its later results on the testing subsets are recorded in Table 4 along with an added metric R^2.

The proposed model's effectiveness may be observed from the good performance it exhibits over other networks. The metrics are used as an indicator of the models' ability to estimate the RUL well. Table 3 shows the model performance against adapted related works in the 50 cycle input sequence range and 125 R_c. The RMSE and MAE for the proposed model is low on all subsets and its standard deviation amongst the 10 splits is low as well. This indicates that the model is consistent and produces a good score on different cuts of the training data subset.

When looking at Table 4, The proposed model scored a lower RMSE and MAE over all data subsets. The best scores are bolded for ease of viewing. An R^2 score that is closer to 1 exhibits stronger correlation and hence suggests that the predicted values follow the true values more closely. The proposed model had a higher R^2 showing that RUL estimates have a stronger relationship to the true values. The performance of the models on the 4 subset data from both train and test sets correspond to the increasing levels of operating and fault complexity as mentioned in Table 1. FD001 generally has best performance over all models while FD004 scores are not as comparable due to its increased complexity.

These results indicate that the proposed model has strong capability to learn intricate relationships in this prognostic application, in both its 10 fold validation on the train set and the test set. It reflects good results from the model. Not only is its metric better than the other three models but a quick glance back at Table 2 would indicate that it has very good performance. However, its comparative success is limited. While the comparison models have served as a reasonable benchmark, it should be pointed out that these parameters are

not optimally found within the literature and that they are optimal for the proposed model. Using a fixed input sequence at 50 also discards small test inputs which would render its direct comparison with related works insufficient. Nevertheless this study into its potential in longer sequences, or a more optimal network hence indicates positive results. A more detailed look at the experimental results that would supplement your understanding of the data presented may be found in Appendix A.

Table 3. Performance comparison with other ANN models, 10 fold validation on training subsets.

Data	Metric	Model			
		CNN [1]	*LSTM*	*DCNN* [2]	*CNN + ATT*
FD001	RMSE	21.61 ± 8.33	12.10 ± 0.69	12.64 ± 1.00	9.19 ± 0.31
	MAE	16.51 ± 7.11	8.76 ± 0.58	9.84 ± 0.92	6.52 ± 0.25
FD002	RMSE	30.58 ± 7.79	16.88 ± 2.53	16.92 ± 0.32	15.41 ± 0.44
	MAE	24.50 ± 8.19	12.78 ± 2.28	13.10 ± 0.24	11.98 ± 0.41
FD003	RMSE	22.80 ± 0.95	10.64 ± 1.30	13.96 ± 0.88	9.13 ± 0.43
	MAE	17.39 ± 0.61	6.98 ± 1.06	10.78 ± 0.72	6.50 ± 0.33
FD004	RMSE	34.48 ± 7.63	17.00 ± 2.18	17.02 ± 0.33	16.89 ± 0.55
	MAE	28.51 ± 9.20	12.25 ± 1.94	13.15 ± 0.32	12.31 ± 0.38

[1] Adapted CNN from [17]. [2] Adapted DCNN from [24].

Table 4. Performance comparison with other ANN models on testing subsets.

Data	Metric	Model			
		CNN [1]	*LSTM*	*DCNN* [2]	*CNN + ATT*
FD001	RMSE	18.86	14.28	12.24	10.60
	MAE	14.73	10.04	9.39	7.55
	R^2	0.78	0.87	0.90	0.93
FD002	RMSE	23.02	18.37	21.02	14.55
	MAE	16.54	13.93	17.32	12.45
	R^2	0.70	0.80	0.75	0.88
FD003	RMSE	21.16	12.79	13.71	11.71
	MAE	15.99	9.60	11.03	8.88
	R^2	0.70	0.89	0.87	0.91
FD004	RMSE	25.69	19.65	26.77	17.23
	MAE	22.32	14.13	22.28	12.83
	R^2	0.63	0.79	0.60	0.83

[1] Adapted CNN from [17]. [2] Adapted DCNN from [24].

3.1.2. Impact of Proposed Structures

The contributions of the temporal convolutions and then the non-standard attention mechanism used in the proposed model are discussed in this section. All the testing are based on FD001 data subset. To get a fair understanding of how well the convolutions are applied, the performance of the first experiment with three models is summarized in Table 5.

- Model 1 removes the proposed attention mechanism from the model and attempts to measure the aptitude of the convolution layers only.
- Model 2 experiments with the convolution by changing the axis of the filters. While Model 1 convolves over each time series, Model 2 takes a more popular approach by convolving over time steps and learning relations between sensor inputs per time step similar to [17].
- Model 3 introduces the attention mechanism back into the alternative convolution Model 2. This is to observe the effects of the attention in the different setting and might provide insight into the model.

- The last entry is the proposed model and it is used as a reference for the other models that are under review.

Table 5. Model investigations.

Metric	Model						
	Model 1 [a]	*Model 2* [a]	*Model 3* [a]	*Model 4* [b]	*Model 5* [b]	*Model 6* [b]	*Proposed Model*
RMSE	12.39	13.49	12.15	13.51	11.91	11.36	10.60
MAE	9.70	9.95	8.50	10.12	8.60	8.29	7.55
R^2	0.90	0.88	0.90	0.88	0.91	0.92	0.93
Score	205.50	256.30	263.75	313.50	272.45	183.50	183.20

[a] Convolution investigation. [b] Attention investigation.

Comparing Model 1 with the proposed model, it shows that the attention mechanism of the proposed model provides a good level of accuracy improvement.

Model 1 and 2 results show us that convolving on the time step outputs does reduce the accuracy of the model. This suggests that for this prognostic application, learning filters that span the length of the time series is more useful to the output of the mode than learning the immediate relations between sensors of each time step.

Model 3 introduces the attention mechanism back into the alternative convolution Model 2. It is obvious that a good level of improvement in the model accuracy which highly suggests with two experiments that the added attention benefits the model.

Figure 3 shows a visualization of the True RUL labels provided from the test set sorted and plotted against the three Models proposed above together with the proposed architecture. One can see that all results tend to fair better as the RUL approaches 0. The points that are further away contain more error. This is appropriate for prognostic setting as accuracy at the time closer to failure is more valuable than the latter. This plot allows us to visualize the impact of the different mechanisms in the three models.

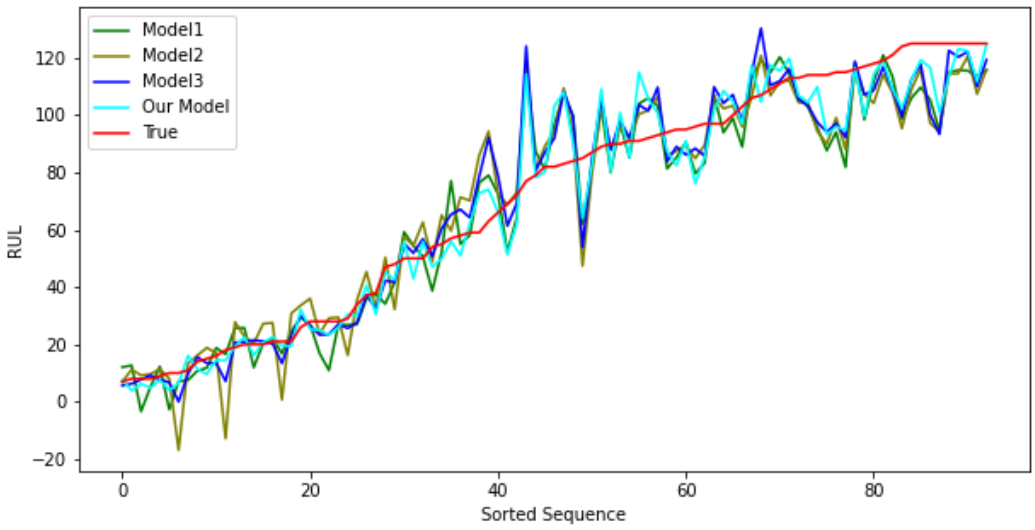

Figure 3. Sorted true labels of FD001 test data subset.

To next get a better understanding of the value of the attention mechanism, a second experiment is conducted with some tweaks centered to the attention mechanism and three more variants are proposed.

- Model 4 sees the attention mechanism attending to the rows of the convolution output instead of the columns. It would thus be focusing between the sensor series instead of amongst the convolution filters.

- Model 5 uses the original Softmax function to generate the attention weights instead of the Sigmoid that is used on top of the alternate attention axis used in Model 4.
- Model 6 uses the proposed model and changes only the Sigmoid to the original Softmax to visualize the difference that the Sigmoid makes.

Table 5 also includes the results of this testing. Model 4 switches the axis of the proposed attention mechanism, attempting to observe the the effectiveness of the mechanism on its chosen vectors. A distinct drop in model accuracy tells us that the attention mechanism is not as capable of capturing important relations amongst the filters between each series. It appears that the original decision of applying the mechanism to the filter outputs was much more effective in this architecture at learning more useful features from the data.

Model 6 is similar to the proposed model except that it uses the Softmax function to calculate the attention weights like in normal attention mechanisms. Model 6 did not fair as well as the proposed model as well. The drop in accuracy is not too significant, but nevertheless present. Hence it is obvious that the Sigmoid function plays a role in improving the the model.

Model 5 gave the most interesting results. Model 5 used the alternate axis for the attention mechanism with the Softmax function. While the two changes added to this model individually decreased accuracy of their models from the proposed method, as can be seen when comparing Model 4 and Model 6 to the proposed method; this model however, did better than Model 4. This suggests that the Softmax function was able to pick out more relevant information from the alternative axis than the Sigmoid that is used in the proposed method. Thus, there are many permutations of different methods that might work especially well with each other.

Figure 4 shows us a visualization of the predicted values of Model 4 to 6 against the proposed model and the true values. The True values and the proposed method are kept the same colour as in Figure 3. Similar observations may be made where RUL predictions for all models closer to 0 are less variant.

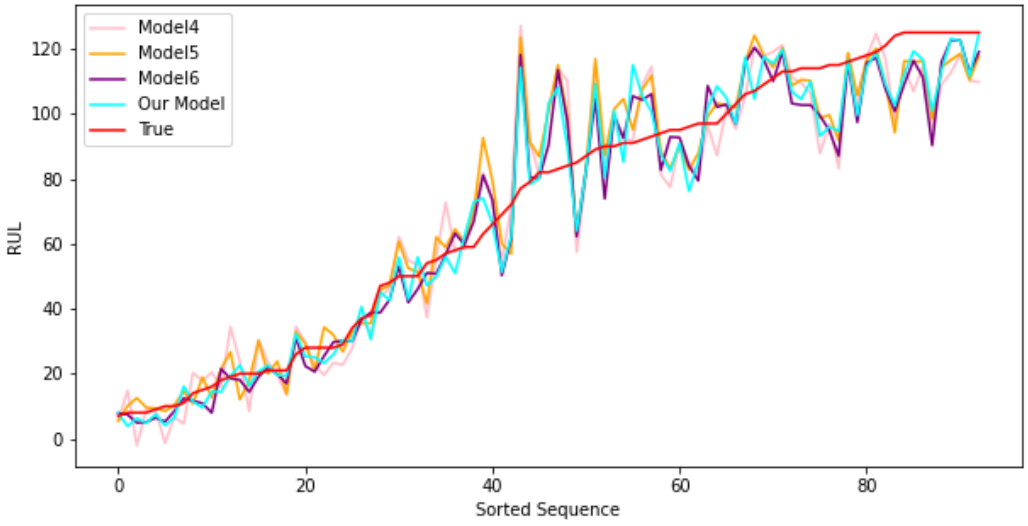

Figure 4. Sorted true labels of FD001 test data subset.

3.2. Hardware Performance

The deployment of all the models and some differences between them are discussed here. Figure 5 shows the parameter differences between the 3 bench-marking models and the proposed models. The proposed model has the second lowest number of parameters at 6299. It is worth noting that the majority of the DCNN parameters are dominated by its dense layer. While the LSTM is large rightfully due to sheer number hidden nodes.

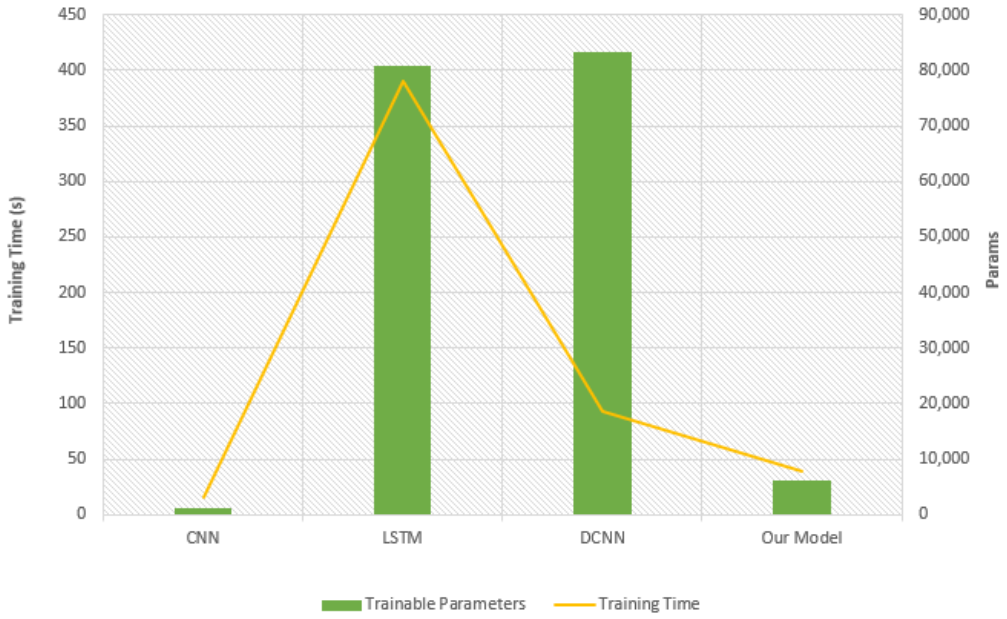

Figure 5. Model parameters and training time for each model.

The training times for the models are also shown in Figure 5. It is obvious that the proposed model here trains faster than the larger models and has comparable training time to the two layer CNN. The LSTM takes the longest to train even though the DCNN has a larger total trainable parameter size. The models are evaluated on a lightweight SoC platform to do a simple benchmark on the models' capabilities in a resource restricted environment. A vanilla Raspberry Pi 3b is used to deploy the models on the same test sets. In the setting of a 1 GB memory constrained platform without a dedicated GPU and only a 1.4 GHz 64 bit quad-core ARM Cortex SoC, the time taken for a prediction is recorded below in Table 6. The values are an average of five measurements over the entire test set. The results are normalized for each model and the time per prediction is calculated and recorded in milliseconds. The times between data subsets are similar for each model. As such, only measurements for FD001 are recorded.

Without the aid of a GPU for efficient floating point matrix multiplication operations nor with a surplus of powerful RAM, the heavier architectures here both fall behind by a large margin. Using only the ARM SoC for deployment of the model, the LSTM was close to 15 times slower than the proposed model and the DCNN was almost 40 times slower. It hence shows that the system is able to handle the proposed architecture much more efficiently.

Table 6. Prediction time on RaspberryPi 3b.

Data	Units: ms			
	CNN [1]	*LSTM*	*DCNN* [2]	*CNN + ATT*
FD001	0.3705	9.2114	22.9548	0.5976

[1] Adapted CNN from [17]. [2] Adapted DCNN from [24].

4. Methods

In this section, the details of the materials and methods used to set up, train and deploy the proposed model are described. The parameters of the proposed model used in this study are reported as well.

Network Setup

The input data from the four data subsets may be clustered. Based on the fault conditions and operating environments, normalization could be done based on those clusters, however for this paper all sensors were normalized individually and while less optimal, no de-noising techniques of the input data was performed and no removal of any input sensors was carried out. The input data comes in with 21 sensor inputs, three settings with an engine ID and time step cycle. All the data, with exception of the 'id', is normalized. The engine id and cycles are used to sort out each engine data into its input sequences.

The piece-wise linear model capped at $R_c = 125$ is applied to the training data after normalization and that would generate the estimated RUL for the training data subsets, getting it ready for supervised training of the model. After this a sliding window the size of the sequence length is propagated down each engine 'id' sequence and generates the data set for training at all instances of the engine health. Sequence length used for comparison with other works in Table 2 are 30, 20, 30, 15. These values cover all inputs from the test and train sets and leave no data sequence out. The sequence length used for investigations is 50 and data sequences that are smaller than this are not used. The ID columns are removed after serving its purpose which leave us with 25 columns of data. No further processing aside from data normalization was done to the inputs.

Table 7 above shows a list of the default parameters used in the proposed architecture. The model is trained with an Adam optimizer with mean square error as loss. The number of epochs vary between data subsets. For FD001 and FD003 that has similar data sizes, 100 epochs are used while for FD002 and FD004 that has a much larger data set, the proposed model is trained for 350 epochs.

Table 7. Parameters of the proposed model.

Parameter	Description	Value
-	Batch size	200
s	Input sequences	25
w_b	Sequence length used in benchmarking	30/20/30/15
w_i	Sequence length used in investigations	50
k	Conv filters	25
c	No. conv layers in H^c	4
-	Conv layer activation	*tanh*
-	Dropout after Attention	0.2
-	Nodes in dense	64
R_c	RUL constant	125

5. Conclusions

With the industries becoming smarter and technology becoming smaller, the data collection techniques will no doubt propagate within manufacturing and maintenance sectors. As more data is made available, the DL models can provide significant value as reliable tools to these industries. In this paper, a novel attention based lightweight CNN is proposed for prognostics application. The experiments were carried out on the C-MAPSS engine degradation data set to show the effectiveness of the proposed model. The objective is to accurately predict the RUL of each engine sequence from raw input sensor data. With the proposed method and parameters selected, the model demonstrated that it can perform well amongst the related works remaining within the top three in every category on this data set. This model is also much smaller and lightweight. It takes significantly less time to train and shows that it can perform in a resource restricted SoC hardware environment. Comparing with the other models, it can be deduced that the model

architecture may be considered for other lightweight requirements and is undoubtedly promising for prognostic applications.

While the competitive experimental results were attained for this method, an investigation into the different mechanisms of the architecture also shows room for greater improvement. It also demonstrated on the cross validated train sets and test sets that an increasing in sequence length can potentially improve the network performance. While the proposed model is only validated on the C-MAPSS data set, a study of its capabilities on other prognostic or MTS data sets and types can be carried out to properly validate the robustness of the architecture. As the network is very small, it likely struggles to learn the features of a more complex tasks or larger set of data. Future research may put more focus into utilizing the attention mechanisms so that a complete multi headed self-attention based architecture for prognostics application may be useful. Such types of attention models show good success in other areas of ML and able to handle more rigorous tasks as well. A more practical approach may also be taken by including the scoring function directly into the training instead of merely as a post reference metric.

Author Contributions: Conceptualization, W.M.T. and T.H.T.; methodology, W.M.T. and T.H.T.; software, W.M.T. and T.H.T.; validation, W.M.T. and T.H.T.; formal analysis, W.M.T. and T.H.T.; investigation, W.M.T. and T.H.T.; resources, T.H.T.; data curation, W.M.T. and T.H.T.; writing—original draft preparation, W.M.T. and T.H.T.; writing—review and editing, T.H.T.; visualization, W.M.T. and; supervision, T.H.T.; project administration, T.H.T.; funding acquisition, T.H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://ti.arc.nasa.gov/c/6/ (accessed on 10 February 2021).

Acknowledgments: The authors would like to thank Wong Thin Sek for his supports to this project.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN Artificial Neural Network
CNN Convolutional Neural Network
DCNN Deep Convolutional Neural Network
DL Deep Learning
DNN Deep Neural Network
LSTM Long Short-Term Memory
SVM Support Vector Machine
ML Machine Learning
MAE Mean Absolute Error
MLP Multi Layer Perceptron
ELM Extreme Learning Machine
DBN Deep Belief Network
MODBNE Multiobjective Deep Belief Networks Ensemble
MTS Multi-variable Time Series
NLP Natural Language Processing
DBN Deep Belief Network
RMSE Root Mean Square Error
RNN Recurrent Neural Network

R^2 R-square correlation
RUL Remaining Useful Life
DAG Directed Acyclic Graph
DSM Deep Survival Model
SoC System on Chip
SVR Support Vector Regressor
TCN Temporal Convolution Network

Appendix A. Further Analysis on Results

While it is apparent from the experiments that our model performs well, it is still useful to look further into the results that are presented and look deeper between the lines for a stronger understanding of its place. The graphs for the four test data subsets may be found below from Figures A1–A4. Each diagram shows the RUL value of each sequence of the test set that was provided by C-MAPSS, sorted in ascending order and represented in red. Each models' corresponding predictions are also displayed. Being able to visualize the data would give a stronger intuition as to how or where a model is performing better. You may notice that Figures A1 and A3 have much less noise than Figures A2 and A4. This is due to the added complexity of FD0002 and FD004. In all instances of this investigation, our model reflects the best performance.

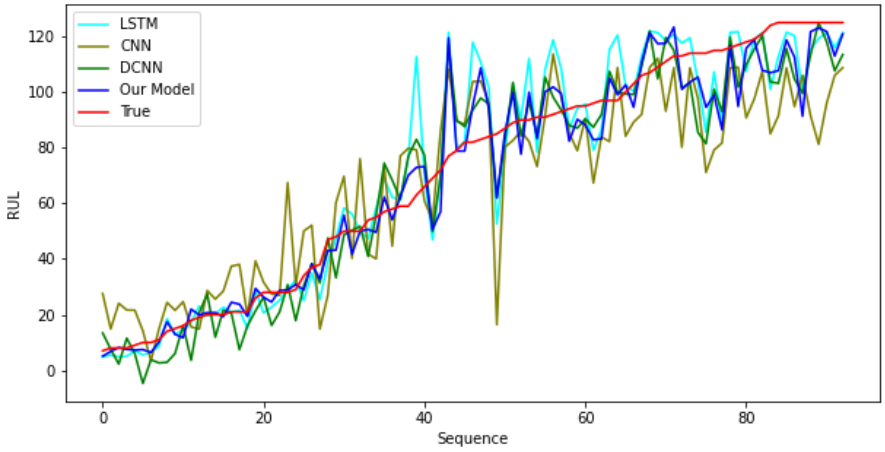

Figure A1. Comparison on FD001 Test set.

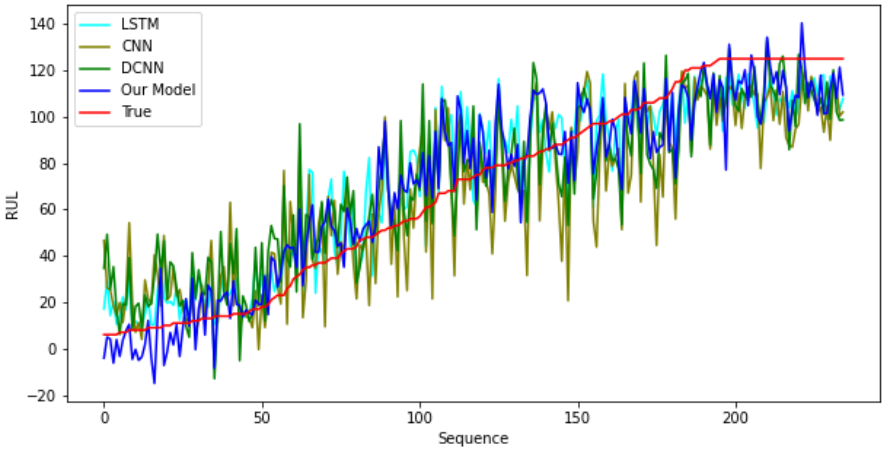

Figure A2. Comparison on FD002 Test set.

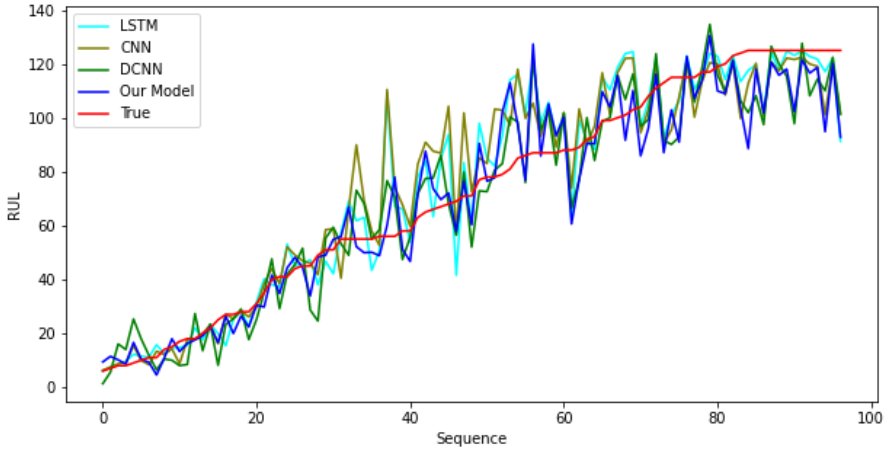

Figure A3. Comparison on FD003 Test set.

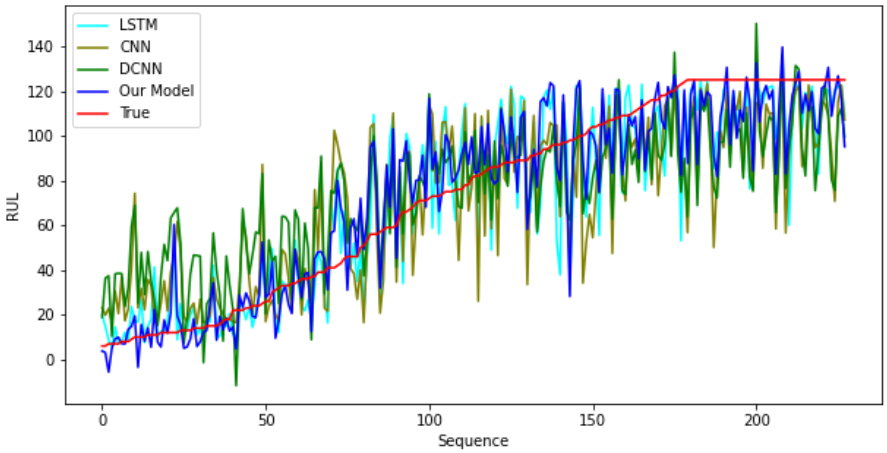

Figure A4. Comparison on FD004 Test set.

The graphs may also be interpreted through Table A1 found below. The table summarizes the metric for each data subset for all the tested models and categories them into its values that correspond to the performance at equal to or less than 10 cycles (C_{10}), 50 cycles (C_{50}) and 100 cycles (C_{100}), respectively. This allows us to access the models performance at different segments of the predicted RUL spectrum. While visible from the graphs, this table quantitatively summarizes the values into its averages for us to better perceive it.

Overall, one might notice that when segmented like this, our model does not outperform in all areas. Specifically, it often performs slightly worse than the LSTM when at 10 cycles or less and at 50 cycles for FD004. This suggests that in some cases, when closer to failure, the LSTM would be more accurate. While an overall score still brings the average best to be our model, it is worth noting that should a situation require higher performance at low cycle counts and disregard cycles larger than that, the LSTM might be the model of choice. For prognostics where the method is largely different between industries and amongst equipment, knowing how or where to apply the best solution would perhaps be the most important.

Table A1. Neural networks performance breakdown.

	Cycles	Metric	Model			
			CNN	*LSTM*	*DCNN*	*Our Model*
FD001	C_{10}	RMSE	10.80	2.59	7.72	2.83
		MAE	9.41	2.29	6.29	2.26
		Mean Score	1.99	0.20	0.78	0.20
	C_{50}	RMSE	13.87	4.33	7.15	3.32
		MAE	10.83	3.35	5.71	2.70
		Mean Score	4.31	0.37	0.73	0.28
	C_{100}	RMSE	17.32	13.31	11.40	9.84
		MAE	13.49	8.86	8.53	6.52
		Mean Score	7.18	4.42	2.52	2.16
FD002	C_{10}	RMSE	15.43	11.28	22.33	11.33
		MAE	12.1	8.71	17.73	8.91
		Mean Score	5.85	2.25	14.46	1.55
	C_{50}	RMSE	16.87	16.54	21.12	12.50
		MAE	12.83	12.19	16.94	9.85
		Mean Score	8.13	10.08	15.53	2.69
	C_{100}	RMSE	21.06	19.89	21.29	14.95
		MAE	16.37	15.28	17.05	11.69
		Mean Score	10.36	13.36	14.77	4.38
FD003	C_{10}	RMSE	11.75	2.31	8.40	3.48
		MAE	10.11	1.70	7.42	2.78
		Mean Score	2.26	0.183	1.23	0.34
	C_{50}	RMSE	17.87	4.75	11.38	4.19
		MAE	13.86	3.41	8.94	3.33
		Mean Score	7.31	0.41	2.39	0.35
	C_{100}	RMSE	23.41	13.15	15.19	10.31
		MAE	18.90	9.16	11.68	7.07
		Mean Score	23.48	4.38	6.43	2.11
FD004	C_{10}	RMSE	23.83	7.30	30.59	5.63
		MAE	20.27	5.64	26.97	4.49
		Mean Score	29.91	0.97	45.01	0.53
	C_{50}	RMSE	22.76	10.98	29.28	12.12
		MAE	17.52	7.56	25.70	8.84
		Mean Score	27.99	2.44	36.27	3.22
	C_{100}	RMSE	24.29	17.83	24.96	16.82
		MAE	19.41	12.96	20.82	12.49
		Mean Score	22.16	7.49	22.91	7.13

Appendix B. Further Details about the NASA C-MAPSS Data Set

More information about this data set may be found in Figure A5 below. The main few rotating components as annotated in the figure are the fan, low pressure compressor, high pressure compressor, high pressure turbine and the low pressure turbine. The outputs include various sensor response surfaces and operability margins. A total of 21 variable sensor outputs out of 58 different outputs available from the model are used in the data set. The outputs may be referenced from Table A2. Though not explicitly treated differently for our use case ANN in the later section, it is useful to know the nature of our sensor outputs.

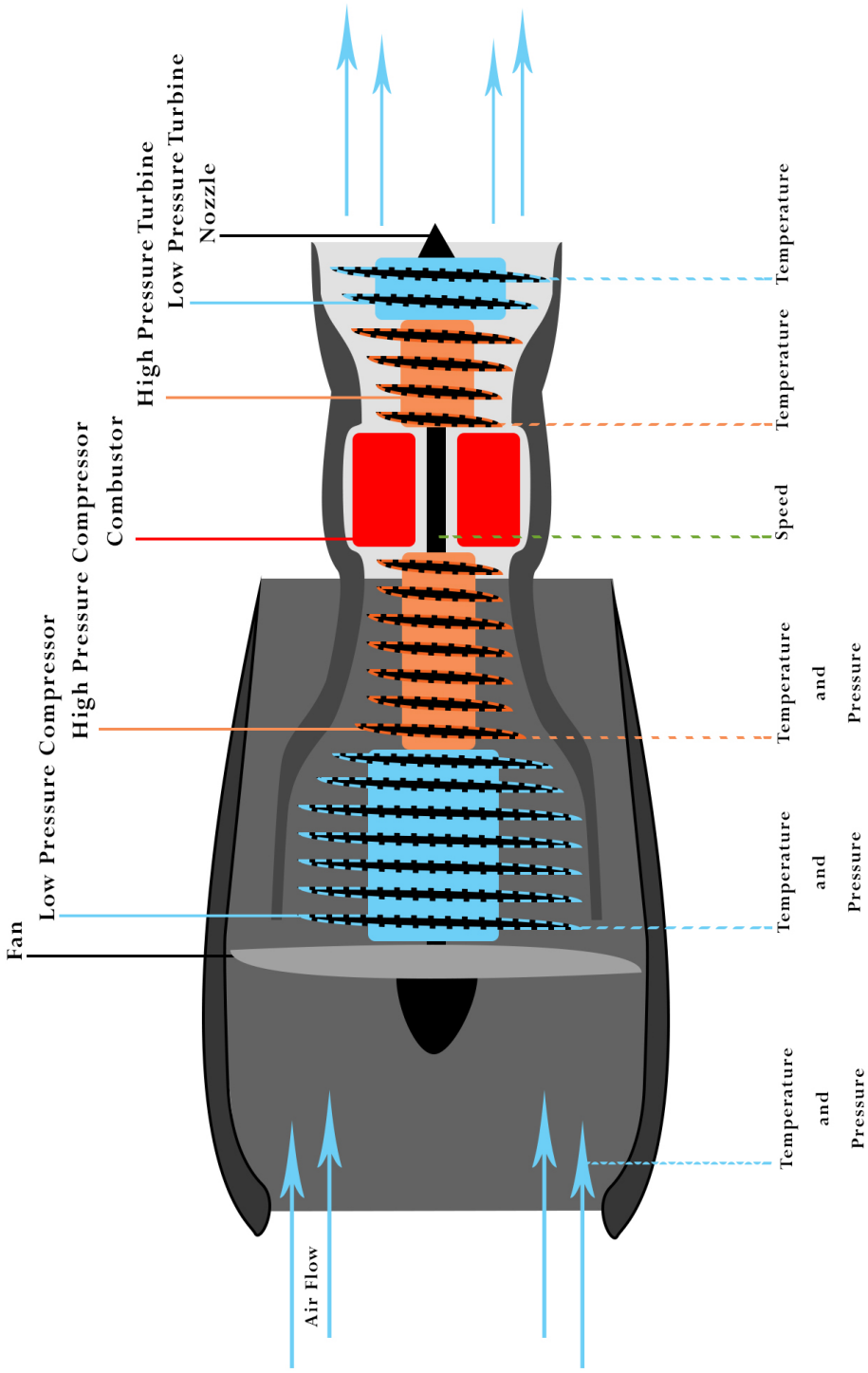

Figure A5. A diagram of Turbofan Engine with its component system of interest and its sensors.

Table A2. Engine sensor outputs from [26].

Symbol	Description	Units
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	–
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	–
farB	Burner fuel-air ratio	–
htBleed	Bleed Enthalpy	–
Nf_{dmd}	Demanded fan speed	rpm
$PCNFR_{dmd}$	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

References

1. Azadeh, A.; Asadzadeh, S.; Salehi, N.; Firoozi, M. Condition-based maintenance effectiveness for series-parallel power generation system—A combined Markovian simulation model. *Reliab. Eng. Syst. Saf.* **2015**, *142*. [CrossRef]
2. Pecht, M.; Gu, J. Physics-of-failure-based prognostics for electronic products. *Trans. Inst. Meas. Control* **2009**, *31*. [CrossRef]
3. Heimes, F. Recurrent Neural Networks for Remaining Useful Life Estimation. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008; pp. 1–6. [CrossRef]
4. van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.W.; Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. *arXiv* **2016**, arXiv:1609.03499.
5. Song, H.; Rajan, D.; Thiagarajan, J.J.; Spanias, A. Attend and Diagnose: Clinical Time Series Analysis using Attention Models. *arXiv* **2017**, arXiv:1711.03905 .
6. Wu, N.; Green, B.; Ben, X.; O'Banion, S. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *arXiv* **2020**, arXiv:2001.08317.
7. Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*; Kremer, S.C., Kolen, J.F., Eds.; IEEE Press: Piscataway, NJ, USA, 2001.
8. Shen, C.H.; Hsu, T.J. Research on Vehicle Trajectory Prediction and Warning Based on Mixed Neural Networks. *Appl. Sci.* **2020**, *11*, 7. [CrossRef]
9. Bai, M.; Liu, J.; Ma, Y.; Zhao, X.; Long, Z.; Yu, D. Long Short-Term Memory Network-Based Normal Pattern Group for Fault Detection of Three-Shaft Marine Gas Turbine. *Energies* **2020**, *14*, 13. [CrossRef]
10. Luong, M.T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. *arXiv* **2015**, arXiv:1508.04025.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
12. Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Łukasz, K. Universal Transformers. *arXiv* **2018**, arXiv:1807.03819.

13. Lai, G.; Chang, W.C.; Yang, Y.; Liu, H. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. *arXiv* **2017**, arXiv:1703.07015.
14. Shih, S.Y.; Sun, F.K.; Lee, H. Temporal Pattern Attention for Multivariate Time Series Forecasting. *arXiv* **2018**, arXiv:1809.04206.
15. Huang, R.; Xi, L.; Li, X.; Liu, C.; Qiu, H.; Lee, J. Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mech. Syst. Signal Process.* **2007**, *21*, 193–207. [CrossRef]
16. Tian, Z. An artificial neural network approach for remaining useful life prediction of equipments subject to condition monitoring. In Proceedings of the 2009 8th International Conference on Reliability, Maintainability and Safety, Chengdu, China, 20–24 July 2009; pp. 143–148.
17. Babu, G.; Zhao, P.; Li, X. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. In Proceedings of the International Conference on Database Systems for Advanced Applications, Dallas, TX, USA, 16–19 April 2016; pp. 214–228. [CrossRef]
18. Javed, K.; Gouriveau, R.; Zerhouni, N. A New Multivariate Approach for Prognostics Based on Extreme Learning Machine and Fuzzy Clustering. *IEEE Trans. Cybern.* **2015**, *45*, 2626–2639. [CrossRef]
19. Zhang, C.; Lim, P.; Qin, A.K.; Tan, K.C. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2306–2318. [CrossRef]
20. Wu, Y.; Yuan, M.; Dong, S.; Lin, L.; Liu, Y. Remaining Useful Life Estimation of Engineered Systems using vanilla LSTM Neural Networks. *Neurocomputing* **2017**, *275*. [CrossRef]
21. Zheng, S.; Ristovski, K.; Farahat, A.; Gupta, C. Long Short-Term Memory Network for Remaining Useful Life estimation. In Proceedings of the 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), Dallas, TX, USA, 19–21 June 2017; pp. 88–95. [CrossRef]
22. Zhang, J.; Wang, P.; Yan, R.; Gao, R. Long short-term memory for machine remaining life prediction. *J. Manuf. Syst.* **2018**. [CrossRef]
23. Wang, J.; Wen, G.; Yang, S.; Liu, Y. Remaining Useful Life Estimation in Prognostics Using Deep Bidirectional LSTM Neural Network. In Proceedings of the 2018 Prognostics and System Health Management Conference (PHM-Chongqing), Chongqing, China, 26–28 October 2018; pp. 1037–1042. [CrossRef]
24. Li, X.; Ding, Q.; Sun, J.Q. Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks. *Reliab. Eng. Syst. Saf.* **2017**, *172*. [CrossRef]
25. Jayasinghe, L.; Samarasinghe, T.; Yuenv, C.; Low, J.C.N.; Ge, S.S. Temporal Convolutional Memory Networks for Remaining Useful Life Estimation of Industrial Machinery. In Proceedings of the 2019 IEEE International Conference on Industrial Technology (ICIT), Melbourne, Australia, 13–15 February 2019; pp. 915–920. [CrossRef]
26. Ma, J.; Su, H.; Zhao, W.l.; Liu, B. Predicting the Remaining Useful Life of an Aircraft Engine Using a Stacked Sparse Autoencoder with Multilayer Self-Learning. *Complexity* **2018**, *2018*, 1–13. [CrossRef]
27. Li, J.; Li, X.; He, D. A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction. *IEEE Access* **2019**, *7*, 75464–75475. [CrossRef]
28. Chu, C.H.; Lee, C.J.; Yeh, H.Y. Developing Deep Survival Model for Remaining Useful Life Estimation Based on Convolutional and Long Short-Term Memory Neural Networks. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 8814658. [CrossRef]
29. Ellefsen, A.; Bjorlykhaug, E.; Æsøy, V.; Ushakov, S.; Zhang, H. Remaining Useful Life Predictions for Turbofan Engine Degradation Using Semi-Supervised Deep Architecture. *Reliab. Eng. Syst. Saf.* **2018**, *183*. [CrossRef]
30. Hou, G.; Xu, S.; Zhou, N.; Yang, L.; Fu, Q. Remaining Useful Life Estimation Using Deep Convolutional Generative Adversarial Networks Based on an Autoencoder Scheme. *Comput. Intell. Neurosci.* **2020**, *2020*, 9601389. [CrossRef]
31. Wang, Z.; Dong, Y.; Liu, W.; Ma, Z. A Novel Fault Diagnosis Approach for Chillers Based on 1-D Convolutional Neural Network and Gated Recurrent Unit. *Sensors* **2020**, *20*, 2458. [CrossRef]
32. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.
33. Teo, T.H.; Tan, W.M.; Tan, Y.S. Tumour detection using Convolutional Neural Network on a lightweight multi-core device. In Proceedings of the 2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (IEEE MCSoC 2019), Singapore, 1–4 October 2019; pp. 87–92.
34. Saxena, A.; Goebel, K.; Simon, D.; Eklund, N. Damage propagation modeling for aircraft engine run-to-failure simulation. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008; pp. 1–9.
35. Saxena, A.; Goebel, K. Turbofan Engine Degradation Simulation Data Set. 2008. Available online: https://ti.arc.nasa.gov/c/6/ (accessed on 10 February 2021).
36. Peel, L. Data driven prognostics using a Kalman filter ensemble of neural network models. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008; pp. 1–6.
37. Ramasso, E. Investigating computational geometry for failure prognostics. *Int. J. Progn. Health Manag.* **2014**, *5*, 1–18.
38. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 1–98. doi:10.1162/NECO_a_00990. [CrossRef] [PubMed]
39. Kim, Y. Convolutional Neural Networks for Sentence Classification. *arXiv* **2014**, arXiv:1408.5882.

40. Cui, Z.; Chen, W.; Chen, Y. Multi-Scale Convolutional Neural Networks for Time Series Classification. *arXiv* **2016**, arXiv:1603.06995.
41. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489. [CrossRef]
42. Li, H.; Min, M.R.; Ge, Y.; Kadav, A. A Context-aware Attention Network for Interactive Question Answering. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017. [CrossRef]
43. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.