Predicting Remaining Useful Life using Time Series Embeddings based on Recurrent Neural Networks

Narendhar Gugulothu, Vishnu TV, Pankaj Malhotra,
Lovekesh Vig, Puneet Agarwal, and Gautam Shroff

TCS Research, New Delhi, India
{narendhar.g, vishnu.tv, malhotra.pankaj}@tcs.com
{lovekesh.vig, puneet.a, gautam.shroff}@tcs.com

Abstract

We consider the problem of estimating the remaining useful life (RUL) of a system or a machine from sensor data. Many approaches for RUL estimation based on sensor data make assumptions about how machines degrade. Additionally, sensor data from machines is noisy and often suffers from missing values in many practical settings. We propose Embed-RUL: a novel approach for RUL estimation from sensor data that does not rely on any degradation-trend assumptions, is robust to noise, and handles missing values. Embed-RUL utilizes a sequence-to-sequence model based on Recurrent Neural Networks (RNNs) to generate embeddings for multivariate time series subsequences. The embeddings for normal and degraded machines tend to be different, and are therefore found to be useful for RUL estimation. We show that the embeddings capture the overall pattern in the time series while filtering out the noise, so that the embeddings of two machines with similar operational behavior are close to each other, even when their sensor readings have significant and varying levels of noise content. We perform experiments on publicly available turbofan engine dataset and a proprietary real-world dataset, and demonstrate that Embed-RUL outperforms the previously reported state-of-the-art (Malhotra, TV, et al., 2016) on several metrics.

1. Introduction

It is quite common in the current era of the 'Industrial Internet of Things' (Da Xu, He, & Li, 2014) for a large number of sensors to be installed for monitoring the operational behavior of machines. Consequently, there is considerable interest in exploiting data from such sensors for health monitoring tasks such as anomaly detection, fault detection, as well as prognostics, i.e., estimating remaining useful life (RUL) of machines in the field.

We highlight some of the practical challenges in using data-driven approaches for health monitoring and RUL estimation, and propose an approach that can handle these challenges:

1) *Health degradation trend*: In complex machines with several components, it is difficult to build physics based models for health degradation analysis. Many data-driven approaches assume a degradation trend, e.g., exponential degradation (Croarkin & Tobias, 2006; Saxena, Goebel, Simon, & Eklund, 2008; Ramasso, 2014; Camci, Eker, Başkan, & Konur, 2016; Wang, Yu, Siegel, & Lee, 2008). This is particularly useful in cases where there is no explicit measurable parameter of the health of a machine. Such an assumption may not hold in other scenarios, e.g., when a component in a machine is approaching failure, the symptoms in the sensor data may initially be intermittent and then grow over time in a non-exponential manner.

2) *Noisy sensor readings*: Sensor readings often suffer from varying levels of environmental noise which entails the use of denoising techniques. The amount of noise may even vary across sensors.

3) *Partial unavailability of sensor data*: Sensor data may be partially unavailable due to several reasons such as network communication loss and damaged or faulty sensors.

4) *Complex temporal dependencies between sensors*: Multiple components interact with each other in a complex way leading to complex dependencies between sensor readings. For example, a change in one sensor may lead to a change in another sensor after a delay of few seconds or even hours. It is desirable to have an approach that can capture the complex operational behavior of machine(s) from sensor readings while accounting for temporal dependencies.

In this paper, we propose *Embed-RUL*: an approach for RUL estimation using Recurrent Neural Networks (RNNs) to address the above challenges. An RNN is used as an encoder to obtain a fixed-dimensional representation that serves as an embedding for multi-sensor time series data. The health of a machine at any point of time can be estimated by comparing an embedding computed using recent sensor history with

representative embeddings computed for periods of normal behavior. Our approach for RUL estimation does not rely on degradation trend assumptions, can handle noise and missing values, and can capture complex temporal dependencies among the sensors. The key contributions of this work are:

- We show that time series embeddings or representations obtained using an RNN Encoder are useful for RUL estimation (refer Section 5.2).

- We show that embeddings are robust and perform well for the RUL estimation task even under noisy conditions, i.e., when sensor readings are noisy (refer Section 5.3).

- Our approach compares favorably to previous benchmarks for RUL estimation (Malhotra, TV, et al., 2016) on the turbofan engine dataset (Saxena & Goebel, 2008) as well as on a real-world pump dataset (refer Section 5.2).

The rest of the paper is organized as follows: We provide a review of related work in Section 2. Section 3 motivates our approach and briefly introduces existing RNN-based approaches for machine health monitoring and RUL estimation using sensor data. In Section 4 we explain our proposed approach for RUL estimation, and provide experimental details and observations in Section 5, and conclude in Section 6.

2. RELATED WORK

Data-driven approaches for RUL estimation: Several approaches for RUL estimation based on sensor data have been proposed. A review of these approaches can be found in (Si, Wang, Hu, & Zhou, 2011). (Eker, Camci, & Jennions, 2014; Khelif, Malinowski, Chebel-Morello, & Zerhouni, 2014) propose estimating RUL directly by calculating the similarity between the sensors without deriving any health estimates. Similarly, Support Vector Regression (Khelif et al., 2017), RNNs (Heimes, 2008), Deep Convolutional Neural Networks (Babu, Zhao, & Li, 2016) have been proposed to estimate the RUL directly by modeling the relations among the sensors without estimating the health of the machines. However, unlike Embed-RUL, none of these approaches focus on robust RUL estimation, and in particular, on robustness to noise.

Robust RUL Estimation: Wavelet filters have been proposed to handle noise for robust performance degradation assessment in (Qiu, Lee, Lin, & Yu, 2003). In (Hu, Youn, Wang, & Yoon, 2012), ensemble of models is used to ensure that predictions are robust. Our proposed approach handles noise in sensor readings by learning robust representations from sensor data via RNN Encoder-Decoder (RNN-ED) models.

Time series representation learning: Unsupervised representation learning for sequences using RNNs has been proposed for applications in various domains including text, video, speech, and time series (e.g., sensor data). Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) based encoders trained using encoder-decoder frame-

work have been proposed to learn representations of video sequences (Srivastava, Mansimov, & Salakhudinov, 2015). Pre-trained LSTM Encoder based on autoencoders are used to initialize networks for classification tasks and are shown to achieve improved performance (Dai & Le, 2015) for text applications. Gated Recurrent Units (GRUs) (Cho et al., 2014) based encoder named Timenet (Malhotra, TV, Vig, Agarwal, & Shroff, 2017) has been recently proposed to obtain embeddings for time series from several domains. The embeddings are shown to be effective for time series classification tasks. Stacked denoising autoencoders have been used to learn hierarchical features from sensor data in (Yan & Yu, 2015). These features are shown to be useful for anomaly detection. However, to the best of our knowledge, the proposed Embed-RUL is the first attempt at using RNN-based embeddings of multivariate sensor data for machine health monitoring, and more specifically, for RUL estimation.

Other Deep learning models for Machine Health Monitoring: Various architectures based on Restricted Boltzmann Machines, RNNs (discussed in Section 3.2) and Convolutional Neural Networks have been proposed for machine health monitoring in different contexts. Many of these architectures and applications of deep learning to machine health monitoring have been surveyed in (Zhao, Yan, et al., 2016). An end-to-end convolutional selective autoencoder for early detection and monitoring of combustion instabilites in high speed flame video frames was proposed in (Akintayo, Lore, Sarkar, & Sarkar, 2016). A combination of deep learning and survival analysis for asset health management has been proposed in (Liao & Ahn, 2016) using sequential data by stacking a LSTM layer, a feed forward layer and a survival model layer to arrive at the asset failure probability. Deep belief networks and autoencoders have been used for health monitoring of aerospace and building systems in (Reddy, Venugopalan, & Giering, 2016). However, none of these approaches are proposed for RUL estimation. Predicting milling machine tool wear from sensor data has been proposed using deep LSTM networks in (Zhao, Wang, Yan, & Mao, 2016). In (Zhao, Yan, Wang, & Mao, 2017), a convolutional bidirectional LSTM network along with fully connected layers at the top is shown to predict tool wear. The convolutional layer extracts robust local features while LSTM layer encodes temporal information. These methods model the problem of degradation estimation in a supervised manner unlike our approach of estimating machine health using embeddings generated using sequence-to-sequence (seq2seq) models (Sutskever, Vinyals, & Le, 2014; Malhotra, TV, et al., 2016).

3. BACKGROUND

Many data-driven approaches attempt to estimate the health of a machine from sensor data in terms of a *health index (HI)*, e.g., (Wang et al., 2008; Ramasso, 2014; Si et al., 2011). The trend of HI over time, referred to as HI curve, is then used

to estimate the RUL by comparing it with the trends of failed instances. The HI curve for a test instance is compared with the HI curve of failed (train) instance to estimate the RUL of the test instance, as shown in Figure 1. In general, the HI curve of the test instance is compared with HI curves of several failed instances, and weighted average of the obtained RUL estimates from the failed instances is used to obtain the final RUL estimate (refer Section 4.3 for details).

In Section 3.1, we introduce a simple approach for HI estimation that maps the current sensor readings to HI. Next, we introduce existing HI estimation techniques that leverage RNNs to capture the temporal patterns in sensor readings, and provide a motivation for our approach in Section 3.2.

3.1. Degradation trend assumption based HI estimation

Consider a HI curve $H = [h_1, h_2, \ldots, h_T]$, where $0 \le h_t \le 1$, $t = 1, 2, \ldots, T$. When a machine is healthy, $h_t = 1$ and when a machine is near failure or about to fail, $h_t = 0$. The multi-sensor readings $\mathbf{x}_t \in \mathbf{R}^n$ at time t can be used to obtain an estimate h'_t for the actual HI value h_t. One way of obtaining this mapping is via a linear regression model: $h'_t = f_{\boldsymbol{\theta}}(\mathbf{x}_t) = \boldsymbol{\theta}^T \mathbf{x}_t + \theta_0$, where $\boldsymbol{\theta} \in \mathbf{R}^n$ and $\theta_0 \in \mathbf{R}$. The parameters $\boldsymbol{\theta}$ and θ_0 are estimated by minimizing $\sum_{t=1}^{T} (h'_t - h_t)^2$, where the *target HI curve* can be assumed to follow an exponential degradation trend, e.g., (Wang et al., 2008).

Once the mapping is learned, the sensor readings at a time instant can be used to obtain HI. Such approaches have two shortcomings: i) they rely on an assumption about the degradation trend, ii) they do not take into account the temporal aspect of the sensor data. We show that the target HI curve for learning such a mapping (i.e., learning the parameters $\boldsymbol{\theta}$ and θ_0) can be obtained using RNN models instead of relying on the exponential assumption (refer Section 5 for details).
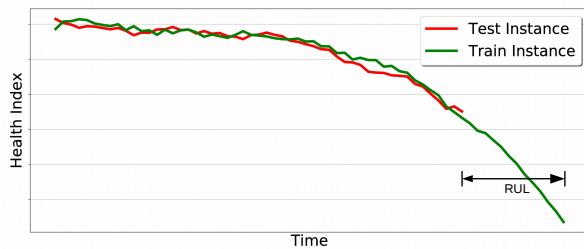

Figure 1. Example of RUL estimation using curve matching.

3.2. RNNs for Machine Health Monitoring

RNNs, especially those based on LSTM units or GRUs have been successfully used to achieve state-of-the-art results on sequence modeling tasks such as machine translation (Cho et al., 2014) and speech recognition (Graves, Mohamed, & Hinton, 2013). Recently, deep RNNs have been shown to

be useful for health monitoring from multi-sensor time series data (Malhotra, Vig, Shroff, & Agarwal, 2015; Malhotra, Ramakrishnan, et al., 2016; Filonov, Lavrentyev, & Vorontsov, 2016). The key idea behind using RNNs for health monitoring is to learn a temporal model of the system by capturing the complex temporal as well as instantaneous dependencies between sensor readings.

Autoencoders have been used to discover interesting structures in the data by means of regularization such as by adding constraints on the number of hidden units of the autoencoder (Ng, 2011), or by adding noise to the input and training the network to reconstruct a denoised version of the input (Vincent, Larochelle, Bengio, & Manzagol, 2008). The key idea behind such autoencoders is that the hidden representation obtained for an input retains the underlying important pattern(s) in the input and ignores the noise component.

RNN autoencoders have been shown to be useful for RUL estimation (Malhotra, TV, et al., 2016) in which the RNN-based model learns to capture the behavior of a machine by learning to reconstruct multivariate time series corresponding to normal behavior in an unsupervised manner. Since the network is trained only on time series corresponding to normal behavior, it is expected to reconstruct the normal behavior well and perform poorly while reconstructing the abnormal behavior. This results in small reconstruction error for normal time series and large reconstruction error for abnormal time series. The reconstruction error is then used as a proxy to estimate the health or degree of degradation, and in turn estimate the RUL of the machine. We refer to this reconstruction error based approach for RUL estimation as *Recon-RUL*.

We propose to learn robust fixed-dimensional representations for multi-sensor time series data via sequence-to-sequence (Sutskever et al., 2014; Bahdanau, Cho, & Bengio, 2014) autoencoders based on RNNs. Here we briefly introduce multilayered RNNs based on GRUs that serve as building blocks of sequence-to-sequence autoencoders (refer Section 4 for details).

3.2.1. Multilayered RNN with Dropout

We use Gated Recurrent Units (Cho et al., 2014) in the hidden layers of sequence-to-sequence autoencoder. Dropout is used for regularization (Pham, Bluche, Kermorvant, & Louradour, 2014; Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) and is applied only to the non-recurrent connections, ensuring information flow across time-steps. For a multilayered RNN with L hidden layers, the hidden state $\mathbf{z}_t^l$ at time t for l^{th} hidden layer is obtained from $\mathbf{z}_{t-1}^l$ and $\mathbf{z}_t^{l-1}$ as in Equation 1- 4. The time series goes through the following transformations iteratively for $t = 1$ through T, where T is length of the time series

$$reset\,gate : \mathbf{r}_t^l = \sigma(\mathbf{W}_r^l \cdot [\mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{z}_{t-1}^l]) \qquad (1)$$
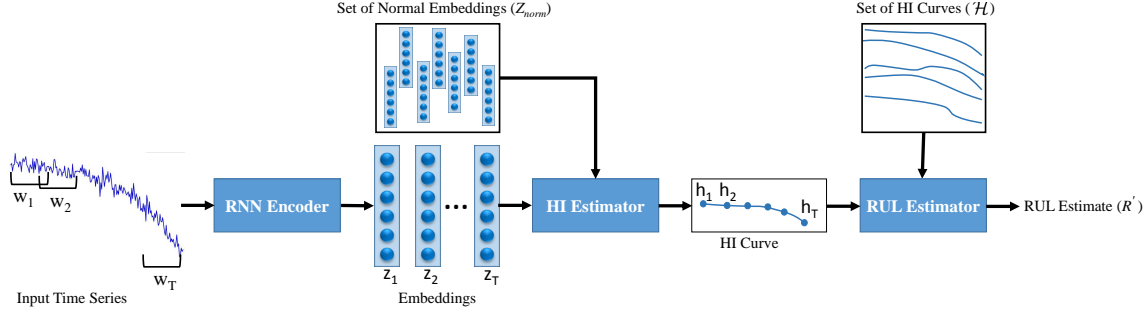
Figure 2. An overview of inference using Embed-RUL. The input time series is divided into windows. Each window is passed through an RNN Encoder to obtain its embedding. The embedding z_t at time t is compared with the embeddings in set Z_{norm} of normal embeddings to obtain health estimate h_t ($t = 1, \ldots, T$). The HI curve is then compared with HI curves of failed train instances in set $\mathcal{H}$ to obtain the RUL estimate R'.

$$update\ gate : \mathbf{u}_t^l = \sigma(\mathbf{W}_u^l \cdot [\mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{z}_{t-1}^l]) \quad (2)$$

$$proposed\ state : \tilde{\mathbf{z}}_t^l = \tanh(\mathbf{W}_p^l \cdot [\mathbf{D}(\mathbf{z}_t^{l-1}), \mathbf{r}_t \odot \mathbf{z}_{t-1}^l]) \quad (3)$$

$$hidden\ state : \mathbf{z}_t^l = (1 - \mathbf{u}_t^l) \odot \mathbf{z}_{t-1}^l + \mathbf{u}_t^l \odot \tilde{\mathbf{z}}_t^l \quad (4)$$

where $\odot$ is Hadamard product, $[\mathbf{a}, \mathbf{b}]$ is concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{D}(\cdot)$ is dropout operator that randomly sets the dimensions of its argument to zero with probability equal to dropout rate, $\mathbf{z}_t^0$ equals the input at time t. $\mathbf{W}_r$, $\mathbf{W}_u$, and $\mathbf{W}_p$ are weight matrices of appropriate dimensions s.t. $\mathbf{r}_t^l, \mathbf{u}_t^l, \tilde{\mathbf{z}}_t^l$, and $\mathbf{z}_t^l$ are vectors in $\mathbf{R}^{c^l}$, where c^l is the number of units in layer l. The sigmoid (σ) and $tanh$ activation functions are applied element-wise.

4. RUL ESTIMATION USING EMBEDDINGS

We consider a scenario where sensor readings over the operational life of one or multiple instances of a machine or a component are available. We denote the set of instances by $\mathcal{I}$. For an instance $i \in \mathcal{I}$, we consider a multi-sensor time series $\mathbf{x}^{(i)} = \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_{T^{(i)}}^{(i)}\}$, where $T^{(i)}$ is the length of the time series, $\mathbf{x}_t^{(i)} \in \mathbf{R}^n$ is an n-dimensional vector corresponding to the readings of the n sensors at time t. For a failed instance i, the length $T^{(i)}$ corresponds to the total operational life (from start to end of life) while for a currently operating instance the length $T^{(i)}$ corresponds to the elapsed operational life till the latest available sensor reading.

Typically, if $T^{(i)}$ is large, we divide the time series into windows (subsequences) of fixed length w. We denote a time series window from time t_1 to t_2 for instance i by $\mathbf{x}^{(i)}(t_1, t_2)$. A fixed-dimensional representation or embedding for each such window is obtained using an RNN Encoder that is trained in an unsupervised manner using RNN-ED. We train RNN-ED using time series subsequences from the entire operational life of machines (including normal as well as faulty operations)[1]. We use the embedding for a window to estimate the

health of the instance at the end of that window. The RNN Encoder is likely to retain the important characterstics of machine behavior in the embeddings, and therefore discriminate between embeddings of windows corresponding to degraded behavior from those of normal behavior. We describe how these embeddings are obtained in Section 4.1, and then describe how health index curves and RUL estimates can be obtained using the embeddings in Sections 4.2 and 4.3, respectively. Figure 2 provides an overview of the steps involved in the proposed approach for RUL estimation.

4.1. Obtaining Embeddings using RNN Encoder-Decoder

We briefly introduce RNN Encoder-Decoder (RNN-ED) networks based on seq2seq learning framework. In general, a seq2seq model consists of a pair of multilayered RNNs trained together: an encoder RNN and a decoder RNN. Figure 3 shows the workings of encoder-decoder pair for a sample time series $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$. Given an input time series $\mathbf{x}^{(i)}(t-w+1, t)$, the encoder RNN iterates through the points in the time series to compute the final hidden state $\mathbf{z}_t^{(i)}$, given by the concatenation of the hidden state vectors from all the layers in the encoder, s.t. $\mathbf{z}_t^{(i)} = [\mathbf{z}_{t,1}^{(i)}, \mathbf{z}_{t,2}^{(i)}, \ldots, \mathbf{z}_{t,L}^{(i)}]$, where $\mathbf{z}_{t,l}^{(i)}$ is the hidden state vector for the lth layer of encoder. The total number of recurrent units in the encoder is given by $c = \sum_{l=1}^{L} c^l$, s.t. $\mathbf{z}_t^{(i)} \in \mathbf{R}^c$ (refer Section 3.2.1).
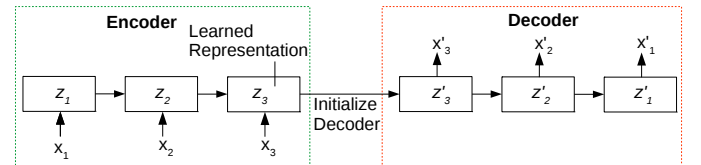

Figure 3. RNN Encoder Decoder for toy time series $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$.

The decoder RNN has the same network structure as the en-

---

[1]Unlike the proposed approach, Recon-RUL (Malhotra, TV, et al., 2016) uses time series subsequences only from normal operation of the machine.

coder, and uses the hidden state $\mathbf{z}_t^{(i)}$ as its initial hidden state, and iteratively (for w steps) goes through the transformations in Equation 1- 4 (followed by a linear output layer) to reconstruct the input time series. The overall process can be thought of as a non-linear mapping of the input multivariate time series to a fixed-dimensional vector representation (embedding) via an encoder function f_{enc}, followed by another non-linear mapping of the fixed-dimensional vector to a multivariate time series via a decoder function f_{dec}:

$$embedding \quad \mathbf{z}_t^{(i)} = f_{enc}(\mathbf{x}^{(i)}(t-w+1,t))$$
$$reconstructed \ time \ series \quad \mathbf{x}'^{(i)}(t-w+1,t) = f_{dec}(\mathbf{z}_t^{(i)}) \tag{5}$$

The reconstruction error at any point t' in $(t-w+1),\ldots,t$ is $\mathbf{e}_{t'}^{(i)} = \mathbf{x}_{t'}^{(i)} - \mathbf{x}_{t'}'^{(i)}$. The overall reconstruction error for the input time series window $\mathbf{x}^{(i)}(t-w+1,t)$ is given by $e_t^{(i)} = \sum_{t'=(t-w+1)}^{t} \| \mathbf{e}_{t'}^{(i)} \|_2^2$. The RNN-ED is trained to minimize the loss function given by the squared reconstruction error: $\mathcal{L} = \sum_{i\in\mathcal{I}} \sum_{t=w}^{T^{(i)}} e_t^{(i)}$.

Typically, along with the final hidden state, an additional input is given to the decoder RNN at each time step. This input is the output of the decoder RNN at the previous time step, as used in (Malhotra, TV, et al., 2016). We, however, do not give any such additional inputs to the decoder along with the final hidden state of encoder. This ensures that the final hidden state of encoder retains all the information required to reconstruct the time series back via the decoder RNN. This approach of learning robust embeddings or representations for time series has been shown to be useful for time series classification in (Malhotra et al., 2017). Figure 4 shows a typical example of input and output from RNN-ED, where the smoothed reconstruction suggests that the embeddings capture the necessary pattern in the input and remove noise.
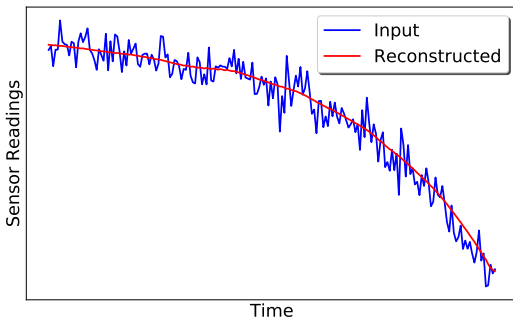

Figure 4. Motivating example showing input time series and its reconstruction using RNN-ED. Sample taken from results on Turbofan Engine dataset (Section 5.2.1).

4.1.1. Handling missing values

In real-world data, the sensor readings tend to be intermittently missing. We include *masking* and *delta* vectors as additional inputs to the RNN-ED at each time instant, as in (Che, Purushotham, Cho, Sontag, & Liu, 2016). The masking vector helps to identify the sensors that are missing at time t, and the delta vector indicates the time elapsed till t, from the most recent non-missing values for sensors in the past. We omit superscript (i) for denoting an instance of the machine from the notation of masking and delta vectors defined below for simplicity.

- *Masking vector* ($\mathbf{m}_t$) denotes the missing sensors at time t and $\mathbf{m}_t \in \{0,1\}^n$, where n is the number of sensors. The jth element of vector $\mathbf{m}_t$ is given by:

$$m_t^j = \begin{cases} 0, & \text{if } x_t^j \text{ is missing} \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where $j = 1,\ldots,n$, and x_t^j denotes the jth element of vector $\mathbf{x}_t$. When $m_t^j = 0$, we set x_t^j to 0 or to the average value for j^{th} sensor (we use 0 for the experiments in Section 5).

- *Delta vector* ($\boldsymbol{\delta}_t$) indicates the time elapsed till t, from the most recent non-missing values for the sensors in the past and $\boldsymbol{\delta}_t \in \mathbf{R}^n$. The jth element of vector $\boldsymbol{\delta}_t$ is given by:

$$\delta_t^j = \begin{cases} y_t - y_{t-1} + \delta_{t-1}^j, & \text{if } t > 1, m_{t-1}^j = 0 \\ y_t - y_{t-1}, & \text{if } t > 1, m_{t-1}^j = 1 \\ 0, & \text{for } t = 1 \end{cases} \tag{7}$$

where $j = 1,\ldots,n$ and $y_t \in \mathbf{R}$ is the time elapsed from start when t^{th} reading is available and $y_1 = 0$. It is to be noted that the sensor readings may not be available at regular time intervals. Therefore, the sequence of readings is indexed by time $t = 1,2,\ldots,T$, while the actual timestamps are denoted by $y_1, y_2, \ldots, y_T$.

The masking and delta vectors are given as additional inputs to the RNN-ED but are not reconstructed, s.t. only the actual sensors are reconstructed. Therefore, the modified input time series $\hat{\mathbf{x}}_t^{(i)} = [\mathbf{x}_t^{(i)}, \mathbf{m}_t^{(i)}, \boldsymbol{\delta}_t^{(i)}]$, while the corresponding target to be reconstructed is $\mathbf{x}_t^{(i)}$. The loss function ($\mathcal{L}$) of the RNN-ED is also modified accordingly, so that the model is not penalized for reconstructing the missing sensors incorrectly. The contribution of a time series subsequence $\mathbf{x}^{(i)}(t-w+1,t)$ to the loss function is thus given by $e_t^{(i)} = \sum_{t'=(t-w+1)}^{t} \| \mathbf{e}_{t'}^{(i)} \cdot \mathbf{m}_{t'}^{(i)} \|_2$. In effect, the network focuses on reconstructing the available sensor readings only.

4.2. Obtaining HI Curves using Embeddings

Here we describe how the embeddings of time series subsequences are utilized to estimate the health of machines. Since

the RNN Encoder captures the important patterns in the input time series subsequences, the embeddings thus obtained can be used to differentiate between normal and degraded regions in the data. We maintain a set of embeddings, Z_{norm}, corresponding to the time series subsequences from the normal behavior of all the instances in $\mathcal{I}$. As a machine operates, its health degrades over time and the corresponding subsequence embeddings tend to be different from those in Z_{norm}. So, we estimate the HI for a subsequence as follows:

$$h_t^{(i)} = min(\| \mathbf{z}_t^{(i)} - \mathbf{z} \|_2), \ \forall \ \mathbf{z} \in Z_{norm} \qquad (8)$$

The HI curve for an instance i obtained from the HI estimates at each time is denoted by $h^{(i)} = \{h_w^{(i)}, h_{w+1}^{(i)}, \ldots, h_{T^{(i)}}^{(i)}\}$. Like the set of normal embeddings Z_{norm}, we also maintain a set $\mathcal{H}$ containing the HI curves of all instances in $\mathcal{I}$.

It is to be noted that HI values are usually assumed to have value between 0 and 1, where 0 means very poor health and 1 means perfect normal health (as shown in Figure 1). The HI as defined in Equation 8 follows inverse definition, i.e. it is low for normal health and high for poor health (as shown in Figure 9). This can be easily transformed to adhere to the standard range of 0-1 through suitable normalization/scaling procedure if required, as used in (Malhotra, TV, et al., 2016).

4.3. RUL Estimation using HI Curves

We use the same approach for estimating RUL from the HI curve as in (Malhotra, TV, et al., 2016). We present it here for the sake of completeness. To estimate the RUL for a test instance i^*, its HI curve $h^{(i^*)}$ is compared with the HI curves in $\mathcal{H}$. The initial health of a train instance and a test instance need not be same. We therefore allow for a time-lag t_D in comparing the HI curve of test instance and train instance.

The similarity between the HI curves of the test instance i^* and a train instance $i \in \mathcal{I}$ for a time-lag t_D is given by:

$$s(i^*, i, t_D) = exp(-\frac{1}{T^{(i^*)}} \sum_{k=w}^{T^{(i^*)}} (h_k^{(i^*)} - h_{k+t_D}^{(i)})^2 / \lambda) \quad (9)$$

$\lambda > 0, t_D \in \{1, 2, ..., \tau\}, t_D + T^{(i^*)} \leq T^{(i)}$. Here, τ is maximum allowed time-lag, and λ controls the notion of similarity: a small value of λ would imply a large value for s even when the difference in HI curves is small. The RUL estimate for i^* based on the HI curve for i and time-lag t_D is given by $R'^{(i^*)}(i, t_D) = T^{(i)} - T^{(i^*)} - t_D$.

A weighted average of the RUL estimates obtained using all combinations of i and t_D is used as the final estimate $R'^{(i^*)}$, and is given by:

$$R'^{(i^*)} = \frac{\sum s(i^*, i, t_D) \cdot R'^{(i^*)}(i, t_D)}{\sum s(i^*, i, t_D)} \qquad (10)$$

where the summation is over only those combinations of i and t_D which satisfy $s(i^*, i, t_D) \geq \alpha.s_{max}$, where $s_{max} = max_{i \in \mathcal{I}, t_D \in \{1, ..., \tau\}} \{s(i^*, i, t_D)\}, 0 \leq \alpha \leq 1$.

5. EXPERIMENTAL EVALUATION

We evaluate our proposed approach for RUL estimation on two datasets: i) a publicly available C-MAPSS Turbofan Engine dataset (Saxena & Goebel, 2008), ii) a proprietary real-world pump dataset. We use Tensorflow (Abadi, Agarwal, Barham, Brevdo, et al., 2016) library for implementing the various RNN models. We present the details of datasets in Section 5.1. In Section 5.2, we show that the results for embedding distance based approaches for RUL estimation compare favorably to the previously reported results using reconstruction error based approaches (Malhotra, TV, et al., 2016) on the engine dataset , as well as on the real-world pump dataset. Further, we evaluate the robustness of the embedding distances and reconstruction error based approaches by measuring the effect of additive random Gaussian noise in the sensor readings on RUL estimation in Section 5.3.

5.1. Datasets Description

5.1.1. Engine dataset

We use the first dataset from the four simulated turbofan engine datasets from the NASA Ames Prognostics Data Repository (Saxena & Goebel, 2008). This dataset contains time series of readings for 24 sensors for 100 train instances (*train_FD001.txt*) of turbofan engine from the beginning of usage till end of life. There are 100 test instances for which the time series are pruned some time prior to failure, s.t. the instances are currently operational and their RUL needs to be estimated (*test_FD001.txt*). The actual RUL for the test instances are available in *RUL_FD001.txt*. Noticeably, each engine instance has a different initial degree of wear such that the initial HI of each instance is likely to be different (implying potential usefulness of τ as introduced in Section 4.3).

We randomly select 80 train instances to train the models. Remaining 20 instances from the train set are used as validation set to select the parameters. The trajectories for these 20 engines are randomly truncated at five different locations to obtain five different instances from each instance for the RUL estimation task. We use Principal Components Analysis (PCA) (Jolliffe, 2002) to reduce the dimensionality of data and select the number of principal components (p) to be used based on the validation set.

5.1.2. Pump dataset

This dataset contains hourly sensor readings for 38 pumps that have reached end of life and 24 pumps that are currently operational. This dataset contains readings over a period of 2.5 years with each pump having 7 sensors installed on it.

The 38 failed instances are randomly split into training, validation and test sets with 70%, 15%, and 15% instances in them, respectively. The 24 operational instances are added to training and validation set only for obtaining the RNN-ED model (they are not part of the set $\mathcal{H}$ as their actual RUL is not known). The data is notably sparse with over 45% missing values across sensors. Also, for most pumps the sensor readings are not available from the date of installation but only few months (average 3.5 months) after the installation date. Depending on the time elapsed, the health degradation level when sensor data is first available for each pump varies significantly. The total operational life of the pumps varies from a minimum of 57 days to a maximum of 726 days.

We downsample the time series data from the original one reading per hour to one reading per day. To do this, we use following four statistics for each sensor over a day as derived sensors: minimum, maximum, average, and standard deviation, such that there are 28 (=7×4) derived sensors for each day. Further, using the derived sensors also helps take care of missing values which reduce from 45% for hourly sampling rate data to 33% for daily sampling rate data. We use masking and delta vectors as additional inputs in this case to train RNN-ED models as described in Section 4.1.1, s.t. the final input dimension is 42 (28 for derived sensors, and 7 each for masking and delta vectors). Unlike the engine dataset where RUL is estimated only at the last available reading for each test instance, here we estimate RUL on every third day of operation for each test instance.

A description of the performance metrics used for evaluation is taken from (Saxena, Celaya, et al., 2008) is provided in Appendix A. The hyper-parameters of our model, to be tuned are: number of principal components (p), number of hidden layers for RNN-ED (L), number of units in a hidden layer l (c^l) (we use same number of units in each hidden layer), dropout rate (d), window length (w), maximum allowed time-lag (τ), similarity threshold (α), maximum predicted RUL (R_{max}), and parameter (λ). The window length (w) can be tuned as a hyperparameter but in practice domain knowledge based selection of window length may be effective.

5.2. Embeddings for RUL Estimation

We follow similar evaluation protocol as used in (Malhotra, TV, et al., 2016). To the best of our knowledge, the reconstruction error based model, LR-ED$_2$, reported the best performance for RUL estimation on the engine dataset in terms of timeliness score (refer Appendix B). We compare variants of embedding distance based approach and reconstruction error based approach. We refer to HI curve obtained using the proposed embedding distance based approach as HI$_e$ (refer Section 4.2), and the HI curve obtained using the reconstruction error based approach in (Malhotra, TV, et al., 2016) as HI$_r$. Here, we refer the reconstruction error based LSTM-ED, LR-ED$_1$ and LR-ED$_2$ models reported in (Malhotra, TV, et al., 2016), as Recon-RUL, Recon-LR$_1$, and Recon-LR$_2$, respectively. We compare following models based on RNNs for RUL estimation task:

- *Embed-RUL Vs Recon-RUL*: We compare RUL estimation performance of Embed-RUL that uses HI$_e$ curves and Recon-RUL that uses HI$_r$ curves.

- *Linear Regression models*: We learn a linear regression model (as described in Section 3.1) using normalized health index curves HI$_e$ as target and call it as Embed-LR$_1$. Embed-LR$_2$ is obtained using squared normalized HI$_e$ as target for the linear regression model. Similarly, Recon-LR$_1$ and Recon-LR$_2$ are obtained based on HI$_r$.

- *RNN Regression model*: RNN-based regression model (RNN-Reg.) is directly used to predict RUL similar approach taken in (Heimes, 2008).

5.2.1. Performance on Engine dataset

We use τ_1=13, τ_2=10 as proposed in (Saxena, Goebel, et al., 2008) for this dataset (refer Equations 11-14 in Appendix A). The parameters are obtained using grid search to minimize the timeliness score S (refer Equation 11) on the validation set. The parameters obtained for the best model (Embed-LR$_1$) are $p = 2$, $L = 1$, $c^l = 55$, $d = 0.2$, $w = 30$, $\tau = 30$, $\alpha = 0.95$, $R_{max} = 120$, and $\lambda = 0.005$.

Table 1 shows the performance in terms of various metrics on this dataset. *We observe that each variant of embedding distance based approach perform better than the corresponding variant of reconstruction error based approach in terms of timeliness score S.* Figure 7(a) shows the distribution of errors for Embed-RUL and Recon-RUL models, and Figure 7(b) shows the distribution of errors for the best linear regression models of embedding distance (Embed-LR$_1$) and reconstruction error (Recon-LR$_2$) based approaches. The error ranges for reconstruction error based models are more spread-out (e.g., -70 to +50 for Recon-RUL) than the corresponding embedding distances based models (e.g., -60 to +30 for Embed-RUL), suggesting the robustness of the embedding distances based models. Figure 5 shows the actual RULs and the RUL estimates from Embed-LR$_1$ and Recon-LR$_2$.

5.2.2. Performance on Pump dataset

The parameters are obtained using grid search to minimize the MSE for RUL estimation on the validation set. The parameters for the best model (Embed-RUL) are $L = 1$, $c^l = 390$, $d = 0.3$, $w = 30$, $\tau = 70$, $\alpha = 0.8$, $R_{max} = 150$, and $\lambda = 10$. The MSE and MAE performance metrics for the RUL estimation task are given in Table 2. *The embedding distance based Embed-RUL model performs significantly better than any of the other approaches.* It is $\approx 35\%$ better than the second best model (Recon-RUL). The linear regression (LR)

Table 1. Engine dataset: Performance comparison (Lower value is better for all metrics except A (%).)

Metric	Recon-RUL	Embed-RUL (proposed)	Recon-LR$_1$	Embed-LR$_1$ (proposed)	Recon-LR$_2$	Embed-LR$_2$ (proposed)	RNN-Reg.
S	1263	**810**	477	**219**	256	**232**	352
MSE	546	**456**	288	**155**	**164**	167	219
A(%)	36	**48**	**65**	59	**67**	62	64
MAE	18	**17**	12	**10**	**10**	**10**	11
MAPE^2	**39**	**39**	20	**19**	**18**	19	17
FPR(%)	34	**23**	19	**14**	**13**	15	22
FNR(%)	30	**29**	**16**	27	**20**	23	24

Table 2. Pump dataset: Performance comparison (We only consider MSE and MAE metrics for this dataset as there are no standard τ_1 and τ_2 known for this dataset to compute the other metrics.)

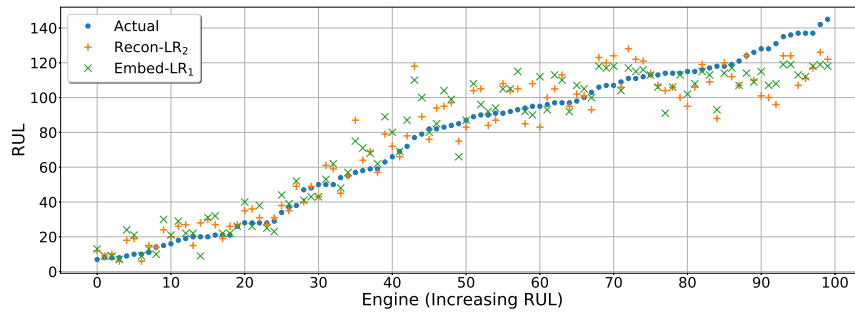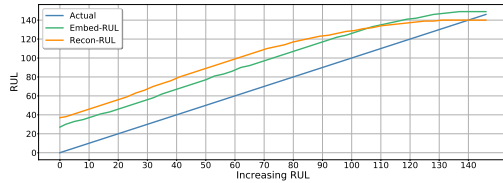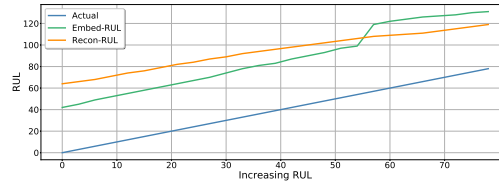| Metric	Recon-RUL	Embed-RUL (proposed)	Recon-LR$_1$	Embed-LR$_1$ (proposed)	Recon-LR$_2$	Embed-LR$_2$ (proposed)	RNN-Reg.
MSE	1979	**1304**	**2277**	2288	2365	**2312**	3422
MAE	40	**33**	**38**	42	**38**	42	48

Figure 5. Engine dataset: RUL estimates given by Embed-LR$_1$ and Recon-LR$_2$.
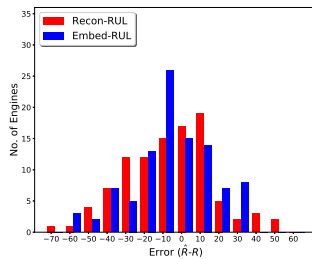

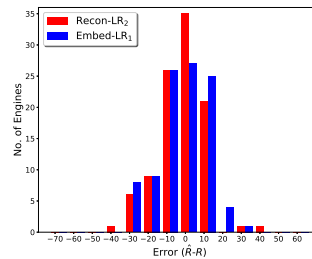(a) Pump for which Embed-RUL MSE is minimum.

(b) Pump for which Embed-RUL MSE is maximum.

Figure 6. Pump dataset: RUL estimates given by Embed-RUL and Recon-RUL.

(a) Embed-RUL Vs Recon-RUL.

(b) Embed-LR$_1$ Vs Recon-LR$_2$.

Figure 7. Engine dataset: Histograms of errors in RUL estimates.

based approaches perform significantly worse than the raw embedding distance or reconstruction error based approaches for HI estimation indicating that the temporal aspect of the sensor readings is very important in this case. Figure 6 shows the actual and estimated RUL values for the pumps with best and worst performance in terms of MSE for the Embed-RUL model.

5.2.3. Qualitative Analysis of Embeddings

We analyze the embeddings given by RNN Encoder for the Embed-RUL models. The original dimension of embeddings for Embed-RUL for engine and pump datasets are 55 and 390, respectively. We use t-SNE (Maaten & Hinton, 2008) to map the embeddings to 2-D space. Figure 8 shows the 2-D scatter plot for the embeddings at the first 25% (normal behavior) and last 25% (degraded behavior) points in the life of all test instances. *We observe that RNN Encoder tends to give different embeddings for windows corresponding to normal and degraded behaviors.* The scatter plots indicate that normal windows are close to each other and far from degraded windows, and vice-versa. Note: Since t-SNE does non-linear dimensionality reduction, the actual distances between normal and degraded windows may not be reflected in these plots.
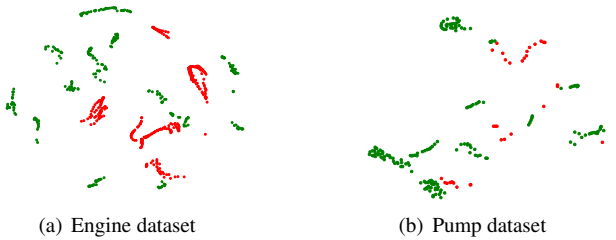

(a) Engine dataset   (b) Pump dataset

Figure 8. t-SNE plot for embeddings of normal (green) and degraded (red) windows.

5.3. Robustness of Embeddings to Noise

We evaluate the robustness of Embed-RUL and Recon-RUL for RUL estimation by adding Gaussian noise to the sensor readings. The sensor reading $\mathbf{x}_t^{(i^*)}$ for a test instance i^* at any time t is corrupted with additive Gaussian noise to obtain a noisy version $\mathbf{x}'^{(i^*)}_t$ s.t. $\mathbf{x}'^{(i^*)}_t | \mathbf{x}_t^{(i^*)} \sim \mathcal{N}(\mathbf{x}_t^{(i^*)}, \sigma^2 I)$.

Table 3 shows the effect of noise on performance for both engine and pump datasets. *For both datasets, the standard deviation of the MSE values over different noise levels is much lesser for Embed-RUL compared to Recon-RUL. This suggests that embedding distance based models are more robust to noise compared to reconstruction error based models.* Also, for the engine dataset, we observe similar behavior in terms of timeliness score (S): 819 ± 41 for Embed-RUL and 1189 ± 110 for Recon-RUL.

Figure 9 depicts a sample scenario showing the health in-

dex generated from noisy sensor data. The vertical bar corresponds to 1-sigma deviation in estimate. The reconstruction error and embedding distance increase over time indicating gradual degradation. While reconstruction error based HI varies significantly with varying noise levels, embedding distance based HI is fairly robust to noise. This suggests that reconstruction error varies significantly with change in noise levels impacting HI estimates while distance between embeddings does not change much leading to robust HI estimates.
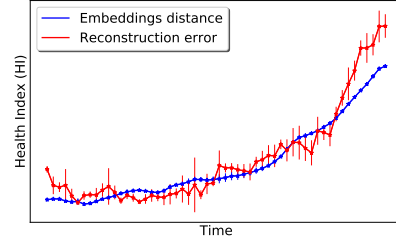

Figure 9. Impact of noise on HI for test engine # 31

Table 3. Robustness Evaluation: MSE values

Noise (σ)	Engine Dataset		Pump Dataset	
	Recon-RUL	Embed-RUL (proposed)	Recon-RUL	Embed-RUL (proposed)
0.0	546	456	1979	1304
0.1	548	462	2003	1298
0.2	521	478	2040	1293
0.3	523	460	2068	1259
0.4	484	473	2087	1280
Mean	524	**466**	2035	**1287**
Std. Dev.	23	**8**	40	**16**

6. DISCUSSION

We have proposed an approach for health monitoring via health index estimation and remaining useful life (RUL) estimation. The proposed approach is capable of dealing with several of the practical challenges in data-driven RUL estimation including noisy sensor readings, missing data, and lack of prior knowledge about degradation trends. The RNN Encoder-Decoder (RNN-ED) is trained in an unsupervised manner to learn fixed-dimensional representations or embeddings to capture machine behavior. The health of a machine is then estimated by comparing the recent embedding with the existing set of embeddings corresponding to normal behavior. We found that our approach using RNN-ED based embedding distances is better compared to the previously known best approach using RNN-ED based reconstruction error on the engine dataset. The proposed approach also gives better results on the real-world pump dataset. We have also shown that embedding distances based RUL estimates are robust to noise.

---

[2]Referred to as MAPE$_1$ in (Malhotra, TV, et al., 2016)

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Akintayo, A., Lore, K. G., Sarkar, S., & Sarkar, S. (2016). Early Detection of Combustion Instabilities using Deep Convolutional Selective Autoencoders on Hi-speed Flame Video. *CoRR*, *abs/1603.07839*. Retrieved from http://arxiv.org/abs/1603.07839

Babu, G. S., Zhao, P., & Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International Conference on Database Systems for Advanced Applications* (pp. 214–228).

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Camci, F., Eker, O. F., Başkan, S., & Konur, S. (2016). Comparison of sensors and methodologies for effective prognostics on railway turnout systems. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, *230*(1), 24–42.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2016). Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Croarkin, C., & Tobias, P. (2006). Nist/sematech e-handbook of statistical methods. *NIST/SEMATECH*.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems* (pp. 3079–3087).

Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, *10*(4), 2233–2243.

Eker, Ö. F., Camci, F., & Jennions, I. K. (2014). A similarity-based prognostics approach for remaining useful life prediction.

Filonov, P., Lavrentyev, A., & Vorontsov, A. (2016). Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *NIPS Time Series Workshop 2016, arXiv preprint arXiv:1612.06676*.

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013* (pp. 6645–6649).

Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. In *Prognostics and Health Management, 2008. PHM 2008.* (pp. 1–6).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hu, C., Youn, B. D., Wang, P., & Yoon, J. T. (2012). Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. *Reliability Engineering & System Safety*, *103*, 120–135.

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library.

Khelif, R., Chebel-Morello, B., Malinowski, S., Laajili, E., Fnaiech, F., & Zerhouni, N. (2017). Direct remaining useful life estimation based on support vector regression. *IEEE Transactions on Industrial Electronics*, *64*(3), 2276–2285.

Khelif, R., Malinowski, S., Chebel-Morello, B., & Zerhouni, N. (2014). Rul prediction based on a new similarity-instance based approach. In *IEEE International Symposium on Industrial Electronics*.

Liao, L., & Ahn, H.-i. (2016). A framework of comnbining deep learning and survival analysis for asset health management. *1st ACM SIGKDD Workshop on ML for PHM.*.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Macmann, O. B., Seitz, T. M., Behbahani, A. R., & Cohen, K. (2016). Performing diagnostics and prognostics on simulated engine failures using neural networks. In *52nd AIAA/SAE/ASEE Joint Propulsion Conference* (p. 4807).

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. In *Anomaly Detection Workshop at 33rd International Conference on Machine Learn-*

ing (ICML 2016), CoRR, abs/1607.00148, 2016, https://arxiv.org/abs/1607.00148.

Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder. *1st ACM SIGKDD Workshop on ML for PHM. arXiv preprint arXiv:1608.06154.*

Malhotra, P., TV, V., Vig, L., Agarwal, P., & Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification. In *25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.*

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 89–94).

Mosallam, A., Medjaher, K., & Zerhouni, N. (2014). Data-driven prognostic method based on bayesian approaches for direct remaining useful life prediction. *Journal of Intelligent Manufacturing.*

Mosallam, A., Medjaher, K., & Zerhouni, N. (2015). Component based data-driven prognostics for complex systems: Methodology and applications. In *Reliability systems engineering (icrse), 2015 first international conference on* (pp. 1–7).

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, *72*(2011), 1–19.

Peng, Y., Wang, H., Wang, J., Liu, D., & Peng, X. (2012). A modified echo state network based remaining useful life estimation approach. In *IEEE Conference on Prognostics and Health Management (PHM).*

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR)* (pp. 285–290).

Qiu, H., Lee, J., Lin, J., & Yu, G. (2003). Robust performance degradation assessment methods for enhanced rolling element bearing prognostics. *Advanced Engineering Informatics*, *17*(3), 127–140.

Ramasso, E. (2014). Investigating computational geometry for failure prognostics. *International Journal of Prognostics and Health Management*, *5*(1), 005.

Reddy, K. K., Venugopalan, V., & Giering, M. J. (2016). Applying deep learning for prognostic health monitoring

of aerospace and building systems. *1st ACM SIGKDD Workshop on ML for PHM..*

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. In *IEEE International Conference on Prognostics and health management. PHM 2008.* (pp. 1–17).

Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository.*

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *IEEE International Conference on Prognostics and Health Management, 2008. PHM 2008.* (pp. 1–9).

Si, X.-S., Wang, W., Hu, C.-H., & Zhou, D.-H. (2011). Remaining useful life estimation–A review on the statistical data driven approaches. *European journal of operational research*, *213*(1), 1–14.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Srivastava, N., Mansimov, E., & Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning* (pp. 843–852).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (pp. 3104–3112).

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096–1103).

Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *IEEE International Conference on Prognostics and Health Management, 2008. PHM 2008.* (pp. 1–6).

Yan, W., & Yu, L. (2015). On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society.*

Zhang, C., Lim, P., Qin, A., & Tan, K. C. (2016). Multi-objective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems.*

Zhao, R., Wang, J., Yan, R., & Mao, K. (2016). Machine health monitoring with LSTM networks. In *IEEE 10th International Conference on Sensing Technology (ICST), 2016* (pp. 1–6).

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2016). Deep Learning and Its Applications to Machine Health Monitoring: A Survey. *arXiv preprint arXiv:1612.07640.*

Zhao, R., Yan, R., Wang, J., & Mao, K. (2017). Learning to monitor machine health with convolutional bi-directional lstm networks. *Sensors*, *17*(2), 273.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long Short-Term Memory Network for Remaining Useful Life estimation. In *IEEE International Conference on Prognostics and Health Management (ICPHM), 2017* (pp. 88–95).

BIOGRAPHIES

Narendhar Gugulothu is a Researcher at TCS Research, New Delhi, India in Data and Decision Sciences group for past 1.5 years. He completed his M.Tech in Communications and Signal processing from Indian Institute of Technology, Hyderabad, India in June, 2016 and B.Tech in Electronics & Communications Engineering from Rajiv Gandhi University of Knowledge Technologies, Basar, India in May, 2014. He has worked in Time series analytics using machine learning and deep learning based models. He has worked on problems like anomaly detection, health monitoring using sensor data from machines in Industrial IOT setting.

Vishnu TV is a Researcher in Data and Decision Sciences group at TCS Research, New Delhi, India for past 2.5 years. He has worked in Time series analytics using machine learning and deep learning based models. He has worked on problems like anomaly detection, health monitoring using sensor data from machines in Industrial IOT setting. He completed his M.Tech in Signal Processing from Indian Institute of Technology Guwahati, India and B.Tech in Electronics and Communications engineering from Jawaharlal Nehru Technological University Anantapur, India.

Pankaj Malhotra is a Researcher in Data and Decision Sciences group at TCS Research, New Delhi, India for past 6 years. He has worked in areas of Big Data Analytics and Time Series Analytics using Machine Learning and Deep Learning based models. He has worked on problems such as customer behavior prediction based on their transaction histories, anomaly detection from sensor data, and health monitoring of complex systems in Industrial IOT setting. Prior to joining TCS Research, he completed his B. Tech. in Electrical Engineering and M. Tech. in Information and Communication Technology from Indian Institute of Technology, New Delhi, India.

Dr. Lovekesh Vig leads the Deep Learning group at TCS Research, New Delhi where he is currently serving as a Senior Scientist. He completed his PhD in Computer Science from Vanderbilt University, USA and has since served as a Financial Engineer at Bloomberg, R&D, New York and as a faculty at the School of Computational and Integrative Sciences, Jawaharlal Nehru University before migrating to TCS Research. His principal research areas are in Robotics, Computational Neuroscience and Neural Networks. Dr. Vig's research group at TCS Research currently works on developing solutions to enterprise problems in Vision, NLP, Robotics and Sensor Analytics.

Puneet Agarwal is a Principal Scientist at Tata Consultancy Services Ltd. He heads Data Analytics and Information Fusion research group at TCS Innovation Labs, Delhi. He has been working in TCS for almost 20 years. His research interests include applied research in natural language processing and data-mining using deep learning. He has extensively worked on time-series and knowledge graph data. He has published many a research papers in various international conferences on Sensor Data Analytics, Natural language processing, Information Fusion, Software Agility, Collaboration, and Model Driven Interpretation. He has a B.E. Degree in Mechanical Engineering from NIT Trichy, India.

Dr. Gautam Shroff is a Vice President and Chief Scientist in Tata Consultancy Services. He heads TCS Research reporting to the Chief Technology Officer of TCS. Prior to joining TCS in 1998, he had been on the faculty of the California Institute of Technology, Pasadena, USA and the Department of Computer Science and Engineering at Indian Institute of Technology, Delhi. Early in his career has

was conferred the Young Scientist Award by the Dept. of Atomic Energy Govt. of India. He has published over 60 papers and two books, serves on the ACM-India Council and the Govt. of India Task Force on AI.

A. PERFORMANCE METRICS

There are several metrics proposed to evaluate the performance of prognostics models (Saxena, Celaya, et al., 2008). We measure the performance of our models in terms of Timeliness Score (S), Accuracy (A), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), False Positive Rate (FPR) and False Negative Rate (FNR) as mentioned in Equations 11-14.

For a test instance i^*, the error $\Delta^{(i^*)} = \hat{R}^{(i^*)} - R^{(i^*)}$ between the estimated RUL ($\hat{R}^{(i^*)}$) and actual RUL ($R^{(i^*)}$). The timeliness score S used to measure the performance of a model is given by:

$$S = \sum_{i^*=1}^{N} (exp(\gamma \cdot |\Delta^{(i^*)}|) - 1) \quad (11)$$

where $\gamma = 1/\tau_1$ if $\Delta^{(i^*)} < 0$, else $\gamma = 1/\tau_2$, N is total test instances. Usually, $\tau_1 > \tau_2$ such that late predictions are penalized more compared to early predictions. The lower the value of S, the better is the performance.

$$A = \frac{100}{N} \sum_{i^*=1}^{N} I(\Delta^{(i^*)}) \quad (12)$$

where $I(\Delta^{(u^*)}) = 1 \, if \, \Delta^{(u^*)} \in [-\tau_1, \tau_2]$, else $I(\Delta^{(u^*)}) = 0$, $\tau_1 > 0, \tau_2 > 0$.

$$MAE = \frac{1}{N} \sum_{i^*=1}^{N} |\Delta^{(i^*)}|, \; MSE = \frac{1}{N} \sum_{i^*=1}^{N} (\Delta^{(i^*)})^2 \quad (13)$$

$$MAPE = \frac{100}{N} \sum_{i^*=1}^{N} \frac{|\Delta^{(i^*)}|}{R^{(i^*)}} \quad (14)$$

A prediction is false positive (FP) if $\Delta^{(i^*)} < -\tau_1$, and false negative (FN) if $\Delta^{(i^*)} > \tau_2$.

B. BENCHMARKS ON TURBOFAN ENGINE DATASET

Various approaches based on Bayesian techniques, e.g., (Mosallam, Medjaher, & Zerhouni, 2014, 2015), Support Vector Regression, e.g., (Khelif et al., 2017), Kalman Filters (Peng, Wang, Wang, Liu, & Peng, 2012), etc. have been proposed. More recently, deep learning models based on Convolutional Neural Networks (CNNs) (Babu et al., 2016), Deep Belief Networks (DBNs) (Zhang, Lim, Qin, & Tan, 2016), and Recurrent Neural Networks (RNNs) (Malhotra, TV, et al., 2016; Zheng, Ristovski, Farahat, & Gupta, 2017; Heimes, 2008) have been proposed. We provide a comparison of some approaches for RUL estimation on the engine dataset (*test_FD001.txt*) below in terms of timeliness score (S) and mean squared error (MSE):

Table 4. Performance of various approaches on Turbofan Engine Data. NR: Not Reported.

Approach	S	MSE
ESN-KF (Peng et al., 2012)	NR	4026
DeepCNN (Babu et al., 2016)	1287	340
SOM[3](Macmann, Seitz, Behbahani, & Cohen, 2016)	NR	297
Shapelet (Khelif et al., 2014)	652	NR
SVR(Khelif et al., 2017)	449	NR
Deep LSTM(Zheng et al., 2017)	338	260
MODBNE(Zhang et al., 2016)	334	226
LR-ED$_2$ (Recon-LR$_2$) (Malhotra, TV, et al., 2016)	256	164
RULCLIPPER[4] (Ramasso, 2014)	216*	176
Embed-LR$_1$(Proposed)	**219**	**155**

---

[3]Dataset simulated under similar settings

[4]Unlike this method which tunes the parameters on the test set to obtain the maximum S, we learn the parameters of the model on a validation set and still get similar performance in terms of S.