# Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder

Pankaj Malhotra, Vishnu TV, Anusha Ramakrishnan
Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, Gautam Shroff
TCS Research, New Delhi, India
{malhotra.pankaj, vishnu.tv, anusha.ramakrishnan}@tcs.com
{gaurangi.anand, lovekesh.vig, puneet.a, gautam.shroff}@tcs.com

## ABSTRACT

Many approaches for estimation of Remaining Useful Life (RUL) of a machine, using its operational sensor data, make assumptions about how a system degrades or a fault evolves, e.g., exponential degradation. However, in many domains degradation may not follow a pattern. We propose a Long Short Term Memory based Encoder-Decoder (LSTM-ED) scheme to obtain an unsupervised health index (HI) for a system using multi-sensor time-series data. LSTM-ED is trained to reconstruct the time-series corresponding to healthy state of a system. The reconstruction error is used to compute HI which is then used for RUL estimation. We evaluate our approach on publicly available Turbofan Engine and Milling Machine datasets. We also present results on a real-world industry dataset from a pulverizer mill where we find significant correlation between LSTM-ED based HI and maintenance costs.

## 1. INTRODUCTION

Industrial Internet has given rise to availability of sensor data from numerous machines belonging to various domains such as agriculture, energy, manufacturing etc. These sensor readings can indicate health of the machines. This has led to increased business desire to perform maintenance of these machines based on their condition rather than following the current industry practice of time-based maintenance. It has also been shown that condition-based maintenance can lead to significant financial savings. Such goals can be achieved by building models for prediction of remaining useful life (RUL) of the machines, based on their sensor readings.

Traditional approach for RUL prediction is based on an assumption that the health degradation curves (drawn w.r.t. time) follow specific shape such as exponential or linear. Under this assumption we can build a model for health index (HI) prediction, as a function of sensor readings. Extrapolation of HI is used for prediction of RUL [29, 24, 25]. However, we observed that such assumptions do not hold in the real-world datasets, making the problem harder to solve. Some of the important challenges in solving the prognostics problem are: i) health degradation curve may not necessarily follow a fixed shape, ii) time to reach same level of degradation by machines of same specifications is often different, iii) each instance has a slightly different initial health or wear, iv) sensor readings if available are

noisy, v) sensor data till end-of-life is not easily available because in practice periodic maintenance is performed.

Apart from the health index (HI) based approach as described above, mathematical models of the underlying physical system, fault propagation models and conventional reliability models have also been used for RUL estimation [5, 26]. Data-driven models which use readings of sensors carrying degradation or wear information such as vibration in a bearing have been effectively used to build RUL estimation models [28, 29, 37]. Typically, sensor readings over the entire operational life of multiple instances of a system from start till failure are used to obtain common degradation behavior trends or to build models of how a system degrades by estimating health in terms of HI. Any new instance is then compared with these trends and the most similar trends are used to estimate the RUL [40].

LSTM networks are recurrent neural network models that have been successfully used for many sequence learning and temporal modeling tasks [12, 2] such as handwriting recognition, speech recognition, sentiment analysis, and customer behavior prediction. A variant of LSTM networks, LSTM encoder-decoder (LSTM-ED) model has been successfully used for sequence-to-sequence learning tasks [8, 34, 4] like machine translation, natural language generation and reconstruction, parsing, and image captioning. LSTM-ED works as follows: An LSTM-based encoder is used to map a multivariate input sequence to a fixed-dimensional vector representation. The decoder is another LSTM network which uses this vector representation to produce the target sequence. We provide further details on LSTM-ED in Sections 4.1 and 4.2.

LSTM Encoder-decoder based approaches have been proposed for anomaly detection [21, 23]. These approaches learn a model to reconstruct the normal data (e.g. when machine is in perfect health) such that the learned model could reconstruct the subsequences which belong to normal behavior. The learnt model leads to high reconstruction error for anomalous or novel subsequences, since it has not seen such data during training. Based on similar ideas, we use Long Short-Term Memory [14] Encoder-Decoder (LSTM-ED) for RUL estimation. In this paper, we propose an unsupervised technique to obtain a health index (HI) for a system using multi-sensor time-series data, which does not make any assumption on the shape of the degradation curve. We use LSTM-ED to learn a model of normal behavior of a system, which is trained to reconstruct multivariate time-series corresponding to normal behavior. The reconstruction error at a point in a time-series is used

to compute HI at that point. In this paper, we show that:

- *LSTM-ED based HI learnt in an unsupervised manner is able to capture the degradation in a system: the HI decreases as the system degrades.*

- *LSTM-ED based HI can be used to learn a model for RUL estimation instead of relying on domain knowledge, or exponential/linear degradation assumption, while achieving comparable performance.*

The rest of the paper is organized as follows: We formally introduce the problem and provide an overview of our approach in Section 2. In Section 3, we describe Linear Regression (LR) based approach to estimate the health index and discuss commonly used assumptions to obtain these estimates. In Section 4, we describe how LSTM-ED can be used to learn the LR model without relying on domain knowledge or mathematical models for degradation evolution. In Section 5, we explain how we use the HI curves of train instances and a new test instance to estimate the RUL of the new instance. We provide details of experiments and results on three datasets in Section 6. Finally, we conclude with a discussion in Section 8, after a summary of related work in Section 7.

## 2. APPROACH OVERVIEW

We consider the scenario where historical instances of a system with multi-sensor data readings till end-of-life are available. The goal is to estimate the RUL of a currently operational instance of the system for which multi-sensor data is available till current time-instance.

More formally, we consider a set of train instances $U$ of a system. For each instance $u \in U$, we consider a multivariate time-series of sensor readings $X^{(u)} = [\mathbf{x}_1^{(u)} \, \mathbf{x}_2^{(u)} \, ... \, \mathbf{x}_{L^{(u)}}^{(u)}]$ with $L^{(u)}$ cycles where the last cycle corresponds to the end-of-life, each point $\mathbf{x}_t^{(u)} \in \mathbf{R}^m$ is an $m$-dimensional vector corresponding to readings for $m$ sensors at time-instance $t$. The sensor data is z-normalized such that the sensor reading $x_{tj}^{(u)}$ at time $t$ for $j$th sensor for instance $u$ is transformed to $\frac{x_{tj}^{(u)} - \mu_j}{\sigma_j}$, where $\mu_j$ and $\sigma_j$ are mean and standard deviation for the $j$th sensor's readings over all cycles from all instances in $U$. (Note: When multiple modes of normal operation exist, each point can normalized based on the $\mu_j$ and $\sigma_j$ for that mode of operation, as suggested in [39].) A subsequence of length $l$ for time-series $X^{(u)}$ starting from time instance $t$ is denoted by $X^{(u)}(t, l) = [x_t^{(u)} \, x_{t+1}^{(u)} \, ... \, x_{t+l-1}^{(u)}]$ with $1 \leq t \leq L^{(u)} - l + 1$.

In many real-world multi-sensor data, sensors are correlated. As in [39, 24, 25], we use Principal Components Analysis to obtain derived sensors from the normalized sensor data with reduced linear correlations between them. The multivariate time-series for the derived sensors is represented as $Z^{(u)} = [\mathbf{z}_1^{(u)} \, \mathbf{z}_2^{(u)} \, ... \, \mathbf{z}_{L^{(u)}}^{(u)}]$, where $\mathbf{z}_t^{(u)} \in \mathbf{R}^p$, $p$ is the number of principal components considered (The best value of $p$ can be obtained using a validation set).

For a new instance $u^*$ with sensor readings $X^{(u^*)}$ over $L^{(u^*)}$ cycles, the goal is to estimate the remaining useful life $R^{(u^*)}$ in terms of number of cycles, given the time-series data for train instances $\{X^{(u)} : u \in U\}$. We describe our approach assuming same operating regime for the entire life of the system. The approach can be easily extended to

multiple operating regimes scenario by treating data for each regime separately (similar to [39, 17]), as described in one of our case studies on milling machine dataset in Section 6.3.

## 3. LINEAR REGRESSION BASED HEALTH INDEX ESTIMATION

Let $H^{(u)} = [h_1^{(u)} \, h_2^{(u)} \, ... \, h_{L^{(u)}}^{(u)}]$ represent the HI curve $H^{(u)}$ for instance $u$, where each point $h_t^{(u)} \in \mathbf{R}$, $L^{(u)}$ is the total number of cycles. We assume $0 \leq h_t^{(u)} \leq 1$, s.t. when $u$ is in perfect health $h_t^{(u)} = 1$, and when $u$ performs below an acceptable level (e.g. instance is about to fail), $h_t^{(u)} = 0$.

Our goal is to construct a mapping $f_{\boldsymbol{\theta}} : \mathbf{z}_t^{(u)} \rightarrow h_t^{(u)}$ s.t.

$$f_{\boldsymbol{\theta}}(\mathbf{z}_t^{(u)}) = \boldsymbol{\theta}^T \mathbf{z}_t^{(u)} + \theta_0 \qquad (1)$$

where $\boldsymbol{\theta} \in \mathbf{R}^p$, $\theta_0 \in \mathbf{R}$, which computes HI $h_t^{(u)}$ from the derived sensor readings $\mathbf{z}_t^{(u)}$ at time $t$ for instance $u$. Given the *target HI curves* for the training instances, the parameters $\boldsymbol{\theta}$ and $\theta_0$ are estimated using Ordinary Least Squares method.

### 3.1 Domain-specific target HI curves

The parameters $\boldsymbol{\theta}$ and $\theta_0$ of the above mentioned Linear Regression (LR) model (Eq. 1) are usually estimated by assuming a mathematical form for the target $H^{(u)}$, with an exponential function being the most common and successfully employed target HI curve (e.g. [10, 33, 40, 29, 6]), which assumes the HI at time $t$ for instance $u$ as

$$h_t^{(u)} = 1 - exp\left(\frac{log(\beta).(L^{(u)} - t)}{(1 - \beta).L^{(u)}}\right), t \in [\beta.L^{(u)}, (1-\beta).L^{(u)}]. \qquad (2)$$

$0 < \beta < 1$. The starting and ending $\beta$ fraction of cycles are assigned values of 1 and 0, respectively.

Another possible assumption is: assume target HI values of 1 and 0 for data corresponding to healthy condition and failure conditions, respectively. Unlike the exponential HI curve which uses the entire time-series of sensor readings, the sensor readings corresponding to only these points are used to learn the regression model (e.g. [38]).

The estimates $\hat{\boldsymbol{\theta}}$ and $\hat{\theta}_0$ based on target HI curves for train instances are used to obtain the *final HI curves* $H^{(u)}$ for all the train instances and a new test instance for which RUL is to be estimated. The HI curves thus obtained are used to estimate the RUL for the test instance based on similarity of train and test HI curves, as described later in Section 5.

## 4. LSTM-ED BASED TARGET HI CURVE

We learn an LSTM-ED model to reconstruct the time-series of the train instances during normal operation. For example, any subsequence corresponding to starting few cycles when the system can be assumed to be in healthy state can be used to learn the model. The reconstruction model is then used to reconstruct all the subsequences for all train instances and the pointwise reconstruction error is used to obtain a target HI curve for each instance. We briefly describe LSTM unit and LSTM-ED based reconstruction model, and then explain how the reconstruction errors obtained from this model are used to obtain the target HI curves.
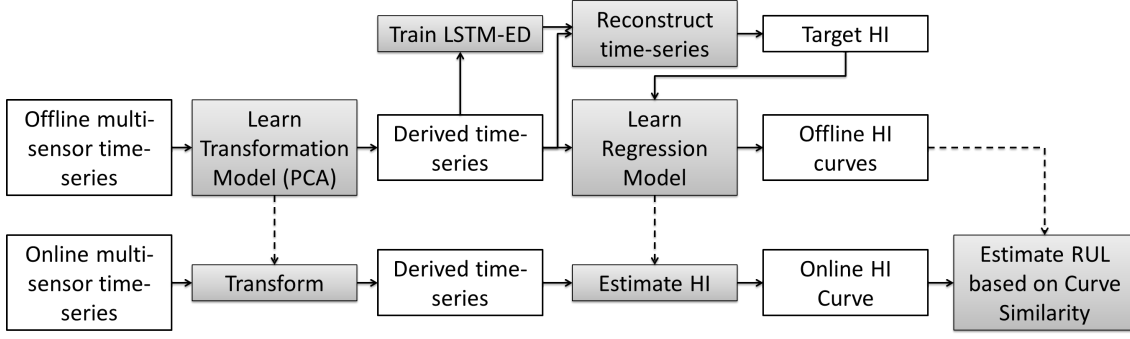
Figure 1: RUL estimation steps using unsupervised HI based on LSTM-ED.

## 4.1 LSTM unit

An LSTM unit is a recurrent unit that uses the input $z_t$, the hidden state activation $a_{t-1}$, and memory cell activation $c_{t-1}$ to compute the hidden state activation $a_t$ at time $t$. It uses a combination of a memory cell $c$ and three types of gates: input gate $i$, forget gate $f$, and output gate $o$ to decide if the input needs to be remembered (using input gate), when the previous memory needs to be retained (forget gate), and when the memory content needs to be output (using output gate).

Many variants and extensions to the original LSTM unit as introduced in [14] exist. We use the one as described in [42]. Consider $T_{n_1,n_2} : \mathbf{R}^{n_1} \to \mathbf{R}^{n_2}$ is an affine transform of the form $\mathbf{z} \mapsto \mathbf{Wz} + \mathbf{b}$ for matrix $\mathbf{W}$ and vector $\mathbf{b}$ of appropriate dimensions. The values for input gate $i$, forget gate $f$, output gate $o$, hidden state $a$, and cell activation $c$ at time $t$ are computed using the current input $z_t$, the previous hidden state $a_{t-1}$, and memory cell value $c_{t-1}$ as given by Eqs. 3-5.

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{m+n,4n} \begin{pmatrix} z_t \\ a_{t-1} \end{pmatrix} \tag{3}$$

Here $\sigma(z) = \frac{1}{1+e^{-z}}$ and $tanh(z) = 2\sigma(2z) - 1$. The operations $\sigma$ and $tanh$ are applied elementwise. The four equations from the above simplifed matrix notation read as: $i_t = \sigma(W_1 z_t + W_2 a_{t-1} + b_i)$, etc. Here, $x_t \in \mathbf{R}^m$, and all others $i_t, f_t, o_t, g_t, a_t, c_t \in \mathbf{R}^n$.

$$c_t = f_t c_{t-1} + i_t g_t \tag{4}$$

$$a_t = o_t tanh(c_t) \tag{5}$$

## 4.2 Reconstruction Model

We consider sliding windows to obtain $L - l + 1$ subsequences for a train instance with $L$ cycles. LSTM-ED is trained to reconstruct the normal (healthy) subsequences of length $l$ from all the training instances. The LSTM encoder learns a fixed length vector representation of the input time-series and the LSTM decoder uses this representation to reconstruct the time-series using the current hidden state and the value predicted at the previous time-step. Given a time-series $Z = [\mathbf{z}_1 \, \mathbf{z}_2 \, ... \, \mathbf{z}_l]$, $\mathbf{a}_t^{(E)}$ is the hidden state of encoder at time $t$ for each $t \in \{1, 2, ..., l\}$, where $\mathbf{a}_t^{(E)} \in \mathbf{R}^c$, $c$ is the number of LSTM units in the hidden layer of the
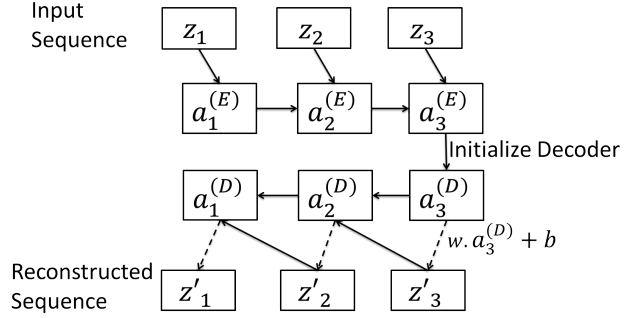


Figure 2: LSTM-ED inference steps for input $\{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$ to predict $\{\mathbf{z}'_1, \mathbf{z}'_2, \mathbf{z}'_3\}$

encoder. The encoder and decoder are jointly trained to reconstruct the time-series in reverse order (similar to [34]), i.e., the target time-series is $[\mathbf{z}_l \, \mathbf{z}_{l-1} \, ... \, \mathbf{z}_1]$. (Note: We consider derived sensors from PCA s.t. $m = p$ and $n = c$ in Eq. 3.)

Fig. 2 depicts the inference steps in an LSTM-ED reconstruction model for a toy sequence with $l = 3$. The value $\mathbf{x}_t$ at time instance $t$ and the hidden state $\mathbf{a}_{t-1}^{(E)}$ of the encoder at time $t - 1$ are used to obtain the hidden state $\mathbf{a}_t^{(E)}$ of the encoder at time $t$. The hidden state $\mathbf{a}_l^{(E)}$ of the encoder at the end of the input sequence is used as the initial state $\mathbf{a}_l^{(D)}$ of the decoder s.t. $\mathbf{a}_l^{(D)} = \mathbf{a}_l^{(E)}$. A linear layer with weight matrix $\mathbf{w}$ of size $c \times m$ and bias vector $\mathbf{b} \in \mathbf{R}^m$ on top of the decoder is used to compute $\mathbf{z}'_t = \mathbf{w}^{\mathbf{T}} \mathbf{a}_t^{(D)} + \mathbf{b}$. During training, the decoder uses $\mathbf{z}_t$ as input to obtain the state $\mathbf{a}_{t-1}^{(D)}$, and then predict $\mathbf{z}'_{t-1}$ corresponding to target $\mathbf{z}_{t-1}$. During inference, the predicted value $\mathbf{z}'_t$ is input to the decoder to obtain $\mathbf{a}_{t-1}^{(D)}$ and predict $\mathbf{z}'_{t-1}$. The reconstruction error $e_t^{(u)}$ for a point $\mathbf{z}_t^{(u)}$ is given by:

$$e_t^{(u)} = \|\mathbf{z}_t^{(u)} - \mathbf{z}'_t^{(u)}\| \tag{6}$$

The model is trained to minimize the objective $E = \sum_{u \in U} \sum_{t=1}^{l} (e_t^{(u)})^2$. It is to be noted that for training, only the subsequences which correspond to perfect health of an instance are considered. For most cases, the first few operational cycles can be assumed to correspond to healthy state for any instance.

## 4.3 Reconstruction Error based Target HI

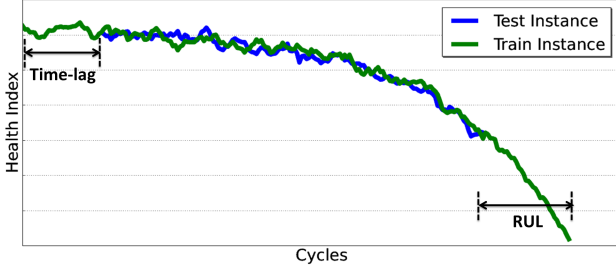A point $\mathbf{z}_t$ in a time-series $Z$ is part of multiple

**Figure 3: Example of RUL estimation using HI curve matching taken from Turbofan Engine dataset (refer Section 6.2)**

overlapping subsequences, and is therefore predicted by multiple subsequences $Z(j,l)$ corresponding to $j = t - l + 1, t - l + 2, ..., t$. Hence, each point in the original time-series for a train instance is predicted as many times as the number of subsequences it is part of ($l$ times for each point except for points $\mathbf{z}_t$ with $t < l$ or $t > L - l$ which are predicted fewer number of times). An average of all the predictions for a point is taken to be final prediction for that point. The difference in actual and predicted values for a point is used as an unnormalized HI for that point.

Error $e_t^{(u)}$ is normalized to obtain the *target HI* $h_t^{(u)}$ as:

$$h_t^{(u)} = \frac{e_M^{(u)} - e_t^{(u)}}{e_M^{(u)} - e_m^{(u)}} \tag{7}$$

where $e_M^{(u)}$ and $e_m^{(u)}$ are the maximum and minimum values of reconstruction error for instance $u$ over $t = 1\ 2\ ...\ L^{(u)}$, respectively. The target HI values thus obtained for all train instances are used to obtain the estimates $\hat{\boldsymbol{\theta}}$ and $\hat{\theta}_0$ (see Eq. 1). Apart from $e_t^{(u)}$, we also consider $(e_t^{(u)})^2$ to obtain target HI values for our experiments in Section 6 such that large reconstruction errors imply much smaller HI value.

# 5. RUL ESTIMATION USING HI CURVE MATCHING

Similar to [39, 40], the HI curve for a test instance $u^*$ is compared to the HI curves of all the train instances $u \in U$. The test instance and train instance may take different number of cycles to reach the same degradation level (HI value). Fig. 3 shows a sample scenario where HI curve for a test instance is matched with HI curve for a train instance by varying the time-lag. The time-lag which corresponds to minimum Euclidean distance between the HI curves of the train and test instance is shown. For a given time-lag, the number of remaining cycles for the train instance after the last cycle of the test instance gives the RUL estimate for the test instance. Let $u^*$ be a test instance and $u$ be a train instance. Similar to [29, 9, 13], we take into account the following scenarios for curve matching based RUL estimation:

1) *Varying initial health across instances*: The initial health of an instance varies depending on various factors such as the inherent inconsistencies in the manufacturing process. We assume initial health to be close to 1. In order to ensure this, the HI values for an instance are divided by the average of its first few HI values (e.g. first 5% cycles). Also, while comparing HI curves $H^{(u^*)}$ and $H^{(u)}$, we allow for a time-lag

$t$ such that the HI values of $u^*$ may be close to the HI values of $H^{(u)}(t, L^{(u^*)})$ at time $t$ such that $t \leq \tau$ (see Eqs. 8-10). This takes care of instance specific variances in degree of initial wear and degradation evolution.

2) *Multiple time-lags with high similarity*: The HI curve $H^{(u^*)}$ may have high similarity with $H^{(u)}(t, L^{(u*)})$ for multiple values of time-lag $t$. We consider multiple RUL estimates for $u^*$ based on total life of $u$, rather than considering only the RUL estimate corresponding to the time-lag $t$ with minimum Euclidean distance between the curves $H^{(u^*)}$ and $H^{(u)}(t, L^{(u^*)})$. The multiple RUL estimates corresponding to each time-lag are assigned weights proportional to the similarity of the curves to get the final RUL estimate (see Eq. 10).

3) *Non-monotonic HI*: Due to inherent noise in sensor readings, HI curves obtained using LR are non-monotonic. To reduce the noise in the estimates of HI, we use moving average smoothening.

4) *Maximum value of RUL estimate*: When an instance is in very good health or has been operational for few cycles, estimating RUL is difficult. We limit the maximum RUL estimate for any test instance to $R_{max}$. Also, the maximum RUL estimate for the instance $u^*$ based on HI curve comparison with instance $u$ is limited by $L^{(u)} - L^{(u^*)}$. This implies that the maximum RUL estimate for any test instance $u$ will be such that the total length $\hat{R}^{(u^*)} + L^{(u^*)} \leq L_{max}$, where $L_{max}$ is the maximum length for any training instance available. Fewer the number of cycles available for a test instance, more difficult it becomes to estimate the RUL.

We define similarity between HI curves of test instance $u^*$ and train instance $u$ with time-lag $t$ as:

$$s(u^*, u, t) = exp(-d^2(u^*, u, t)/\lambda) \tag{8}$$

where,

$$d^2(u^*, u, t) = \frac{1}{L^{(u^*)}} \sum_{i=1}^{L^{(u^*)}} (h_i^{(u^*)} - h_{i+t}^{(u)})^2 \tag{9}$$

is the squared Euclidean distance between $H^{(u^*)}(1, L^{(u^*)})$ and $H^{(u)}(t, L^{(u^*)})$, and $\lambda > 0$, $t \in \{1, 2, ..., \tau\}$, $t + L^{(u^*)} \leq L^{(u)}$. Here, $\lambda$ controls the notion of similarity: a small value of $\lambda$ would imply large difference in $s$ even when $d$ is not large. The RUL estimate for $u^*$ based on the HI curve for $u$ and for time-lag $t$ is given by $\hat{R}^{(u^*)}(u, t) = L^{(u)} - L^{(u^*)} - t$.

The estimate $\hat{R}^{(u^*)}(u, t)$ is assigned a weight of $s(u^*, u, t)$ such that the weighted average estimate $\hat{R}^{(u^*)}$ for $R^{(u^*)}$ is given by

$$\hat{R}^{(u^*)} = \frac{\sum s(u^*, u, t).\hat{R}^{(u^*)}(u, t)}{\sum s(u^*, u, t)} \tag{10}$$

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

It is to be noted that the parameter $\alpha$ decides the number of RUL estimates $\hat{R}^{(u^*)}(u, t)$ to be considered to get the final RUL estimate $\hat{R}^{(u^*)}$. Also, variance of the RUL estimates $\hat{R}^{(u^*)}(u, t)$ considered for computing $\hat{R}^{(u^*)}$ can be used as a measure of confidence in the prediction, which is useful in practical applications (for example, see Section 6.2.2). During the initial stages of an instance's usage, when it is in good health and a fault has still not appeared, estimating

RUL is tough, as it is difficult to know beforehand how exactly the fault would evolve over time once it appears.

## 6. EXPERIMENTAL EVALUATION

We evaluate our approach on two publicly available datasets: C-MAPSS Turbofan Engine Dataset [33] and Milling Machine Dataset [1], and a real world dataset from a pulverizer mill. For the first two datasets, the ground truth in terms of the RUL is known, and we use RUL estimation performance metrics to measure efficacy of our algorithm (see Section 6.1). The Pulverizer mill undergoes repair on timely basis (around one year), and therefore ground truth in terms of actual RUL is not available. We therefore draw comparison between health index and the cost of maintenance of the mills.

For the first two datasets, we use different target HI curves for learning the LR model (refer Section 3): *LR-Lin* and *LR-Exp* models assume linear and exponential form for the target HI curves, respectively. *LR-ED$_1$* and *LR-ED$_2$* use normalized reconstruction error and normalized squared-reconstruction error as target HI (refer Section 4.3), respectively. The target HI values for LR-Exp are obtained using Eq. 2 with $\beta = 5\%$ as suggested in [40, 39, 29].

### 6.1 Performance metrics considered

Several metrics have been proposed for evaluating the performance of prognostics models [32]. We measure the performance in terms of Timeliness Score (S), Accuracy (A), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Absolute Percentage Error (MAPE$_1$ and MAPE$_2$) as mentioned in Eqs. 11-15, respectively. For test instance $u^*$, the error $\Delta^{(u^*)} = \hat{R}^{(u^*)} - R^{(u^*)}$ between the estimated RUL ($\hat{R}^{(u^*)}$) and actual RUL ($R^{(u^*)}$). The score $S$ used to measure the performance of a model is given by:

$$S = \sum_{u^*=1}^{N} \left( exp(\gamma.|\Delta^{(u^*)}|) - 1 \right) \quad (11)$$

where $\gamma = 1/\tau_1$ if $\Delta^{(u^*)} < 0$, else $\gamma = 1/\tau_2$. 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.

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

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

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

$$MAPE_1 = \frac{100}{N} \sum_{u^*=1}^{N} \frac{|\Delta^{(u^*)}|}{R^{(u^*)}} \quad (14)$$

$$MAPE_2 = \frac{100}{N} \sum_{u^*=1}^{N} \frac{|\Delta^{(u^*)}|}{R^{(u^*)} + L^{(u^*)}} \quad (15)$$

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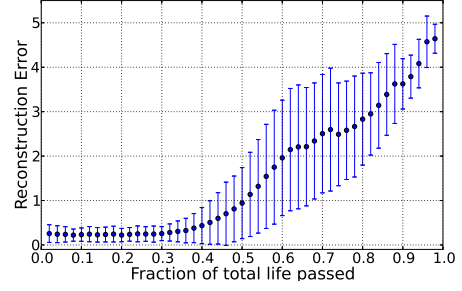


**Figure 4:** Turbofan Engine: Average pointwise reconstruction error based on LSTM-ED w.r.t. fraction of total life passed.

### 6.2 C-MAPSS Turbofan Engine Dataset

We consider the first dataset from the simulated turbofan engine data [33] (NASA Ames Prognostics Data Repository). The dataset contains readings for 24 sensors (3 operation setting sensors, 21 dependent sensors) for 100 engines till a failure threshold is achieved, i.e., till end-of-life in $train\_FD001.txt$. Similar data is provided for 100 test engines in $test\_FD001.txt$ where the time-series for engines are pruned some time prior to failure. The task is to predict RUL for these 100 engines. The actual RUL values are provided in $RUL\_FD001.txt$. There are a total of 20631 cycles for training engines, and 13096 cycles for test engines. Each engine has a different degree of initial wear. We use $\tau_1 = 13, \tau_2 = 10$ as proposed in [33].

#### 6.2.1 Model learning and parameter selection

We randomly select 80 engines for training the LSTM-ED model and estimating parameters $\boldsymbol{\theta}$ and $\theta_0$ of the LR model (refer Eq. 1). The remaining 20 training instances are used for selecting the parameters. The trajectories for these 20 engines are randomly truncated at five different locations s.t. five different cases are obtained from each instance. Minimum truncation is 20% of the total life and maximum truncation is 96%. For training LSTM-ED, only the first subsequence of length $l$ for each of the selected 80 engines is used. The parameters number of principal components $p$, the number of LSTM units in the hidden layers of encoder and decoder $c$, window/subsequence length $l$, maximum allowed time-lag $\tau$, similarity threshold $\alpha$ (Eq. 10), maximum predicted RUL $R_{max}$, and parameter $\lambda$ (Eq. 8) are estimated using grid search to minimize $S$ on the validation set. The parameters obtained for the best model (LR-ED$_2$) are $p = 3$, $c = 30$, $l = 20$, $\tau = 40$, $\alpha = 0.87$, $R_{max} = 125$, and $\lambda = 0.0005$.

#### 6.2.2 Results and Observations

**LSTM-ED based Unsupervised HI**: Fig. 4 shows the average pointwise reconstruction error (refer Eq. 6) given by the model LSTM-ED which uses the pointwise reconstruction error as an unnormalized measure of health (higher the reconstruction error, poorer the health) of all the 100 test engines w.r.t. percentage of life passed (for derived sensor sequences with $p = 3$ as used for model LR-ED$_1$ and LR-ED$_2$). During initial stages of an engine's life, the average reconstruction error is small. *As the number of cycles passed increases, the reconstruction error increases. This suggests that reconstruction error can be used as an*
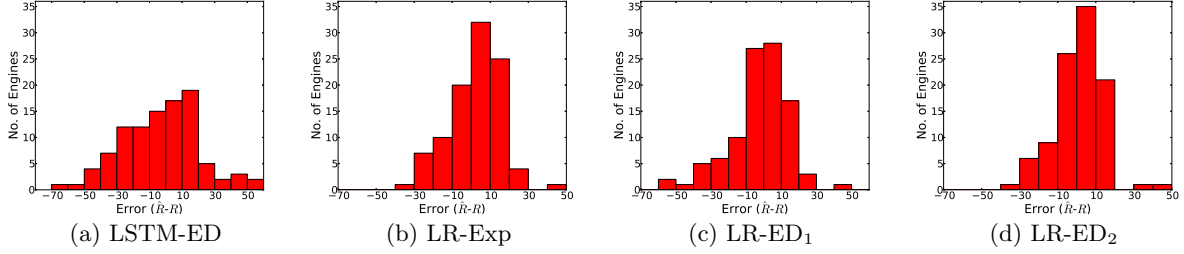
(a) LSTM-ED  (b) LR-Exp  (c) LR-ED$_1$  (d) LR-ED$_2$

Figure 5: Turbofan Engine: Histograms of prediction errors for Turbofan Engine dataset.



(a) RUL estimates given by LR-Exp, LR-ED$_1$, and LR-ED$_2$

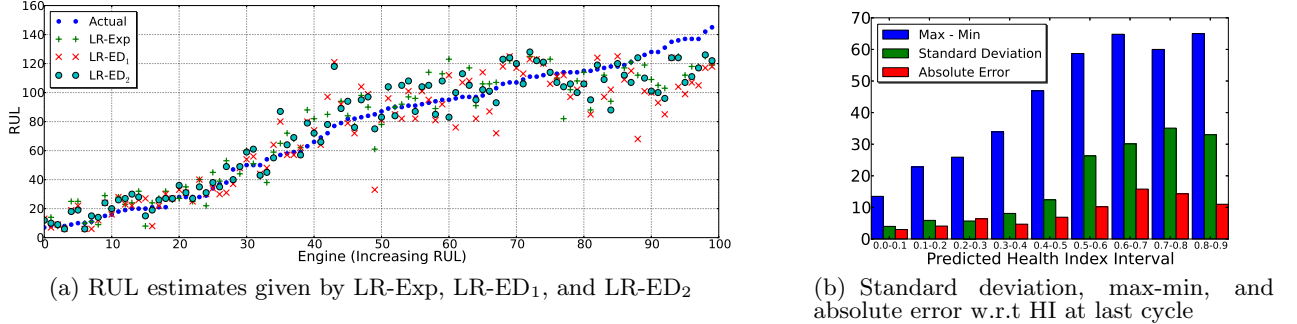(b) Standard deviation, max-min, and absolute error w.r.t HI at last cycle

Figure 6: Turbofan Engine: (a) RUL estimates for all 100 engines, (b) Absolute error w.r.t. HI at last cycle

*indicator of health of a machine.* Fig. 5(a) and Table 1 suggest that RUL estimates given by HI from LSTM-ED are fairly accurate. On the other hand, the 1-sigma bars in Fig. 4 also suggest that the reconstruction error at a given point in time (percentage of total life passed) varies significantly from engine to engine.

**Performance comparison**: Table 1 and Fig. 5 show the performance of the four models LSTM-ED (without using linear regression), LR-Exp, LR-ED$_1$, and LR-ED$_2$. We found that LR-ED$_2$ performs significantly better compared to the other three models. LR-ED$_2$ is better than the LR-Exp model which uses domain knowledge in the form of exponential degradation assumption. We also provide comparison with RULCLIPPER (RC) [29] which (to the best of our knowledge) has the best performance in terms of timeliness $S$, accuracy $A$, MAE, and MSE [31] reported in the literature[1] on the turbofan dataset considered and four other turbofan engine datasets (*Note:* Unlike RC, we learn the parameters of the model on a validation set rather than test set.) RC relies on the exponential assumption to estimate a HI polygon and uses intersection of areas of polygons of train and test engines as a measure of similarity to estimate RUL (similar to Eq. 10, see [29] for details). *The results show that LR-ED$_2$ gives performance comparable to RC without relying on the domain-knowledge based exponential assumption.*

The worst predicted test instance for LR-Exp, LR-ED$_1$ and LR-ED$_2$ contributes 23%, 17%, and 23%, respectively, to the timeliness score $S$. For LR-Exp and LR-ED$_2$ it is nearly $1/4^{th}$ of the total score, and suggests that for other 99 test engines the timeliness score $S$ is very good.

---

[1] For comparison with some of the other benchmarks readily available in literature, see Table 4. It is to be noted that the comparison is not exhaustive as a survey of approaches for the turbofan engine dataset since [31] is not available.

|  | LSTM-ED | LR-Exp | LR-ED$_1$ | LR-ED$_2$ | RC |
|---|---|---|---|---|---|
| **S** | 1263 | 280 | 477 | 256 | **216** |
| **A(%)** | 36 | 60 | 65 | **67** | **67** |
| **MAE** | 18 | **10** | 12 | **10** | **10** |
| **MSE** | 546 | 177 | 288 | **164** | 176 |
| **MAPE$_1$(%)** | 39 | 21 | 20 | **18** | 20 |
| **MAPE$_2$(%)** | 9 | 5.2 | 5.9 | **5.0** | NR |
| **FPR(%)** | 34 | **13** | 19 | **13** | 56 |
| **FNR(%)** | 30 | 27 | **16** | 20 | 44 |

Table 1: Turbofan Engine: Performance comparison

**HI at last cycle and RUL estimation error**: Fig. 6(a) shows the actual and estimated RULs for LR-Exp, LR-ED$_1$, and LR-ED$_2$. For all the models, we observe that as the actual RUL increases, the error in predicted values increases. Let $R_{all}^{(u^*)}$ denote the set of all the RUL estimates $\hat{R}^{(u^*)}(u,t)$ considered to obtain the final RUL estimate $\hat{R}^{(u^*)}$ (see Eq. 10). Fig. 6(b) shows the average values of absolute error, standard deviation of the elements in $R_{all}^{(u^*)}$, and the difference of the maximum and the minimum value of the elements in $R_{all}^{(u^*)}$ w.r.t. HI value at last cycle. It suggests that when an instance is close to failure, i.e., HI at last cycle is low, RUL estimate is very accurate with low standard deviation of the elements in $R_{all}^{(u^*)}$. On the other hand, when an instance is in good health, i.e., when HI at last cycle is close to 1, the error in RUL estimate is high, and the elements in $R_{all}^{(u^*)}$ have high standard deviation.

## 6.3  Milling Machine Dataset

This data set presents milling tool wear measurements from a lab experiment. Flank wear is measured for 16 cases with each case having varying number of runs of varying durations. The wear is measured after runs but not necessarily after every run. The data contains readings for 10 variables (3 operating condition variables, 6 dependent
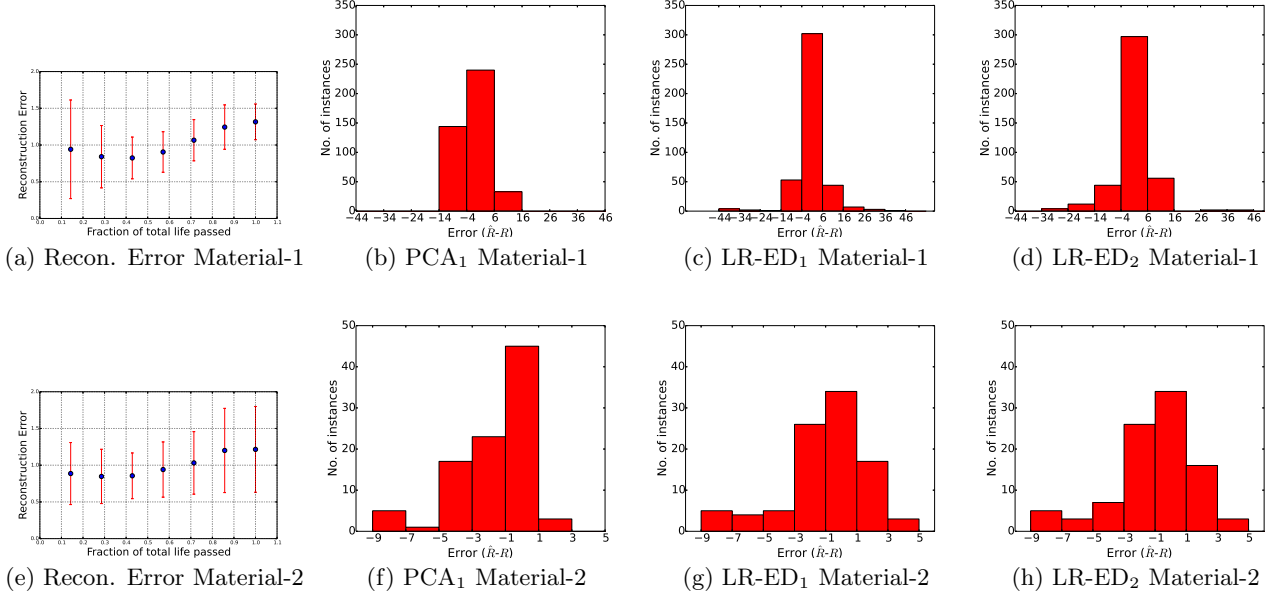
|  (a) Recon. Error Material-1 | (b) PCA$_1$ Material-1 | (c) LR-ED$_1$ Material-1 | (d) LR-ED$_2$ Material-1 |

| (e) Recon. Error Material-2 | (f) PCA$_1$ Material-2 | (g) LR-ED$_1$ Material-2 | (h) LR-ED$_2$ Material-2 |

**Figure 7: Milling Machine: Reconstruction errors w.r.t. cycles passed and histograms of prediction errors for milling machine dataset.**

sensors, 1 variable measuring time elapsed until completion of that run). A snapshot sequence of 9000 points during a run for the 6 dependent sensors is provided. We assume each run to represent one cycle in the life of the tool. We consider two operating regimes corresponding to the two types of material being milled, and learn a different model for each material type. There are a total of 167 runs across cases with 109 runs and 58 runs for material types 1 and 2, respectively. Case number 6 of material 2 has only one run, and hence not considered for experiments.

### 6.3.1 Model learning and parameter selection

Since number of cases is small, we use leave one out method for model learning and parameters selection. For training the LSTM-ED model, the first run of each case is considered as normal with sequence length of 9000. An average of the reconstruction error for a run is used to the get the target HI for that run/cycle. We consider mean and standard deviation of each run (9000 values) for the 6 sensors to obtain 2 derived sensors per sensor (similar to [9]). We reduce the gap between two consecutive runs, via linear interpolation, to 1 second (if it is more); as a result HI curves for each case will have a cycle of one second. The tool wear is also interpolated in the same manner and the data for each case is truncated until the point when the tool wear crosses a value of 0.45 for the first time. The target HI from LSTM-ED for the LR model is also interpolated appropriately for learning the LR model.

The parameters obtained for the best models (based on minimum MAPE$_1$) for material-1 are $p = 1$, $\lambda = 0.025$, $\alpha = 0.98$, $\tau = 15$ for PCA$_1$, for material-2 are $p = 2$, $\lambda = 0.005$, $\alpha = 0.87$, $\tau = 13$, and $c = 45$ for LR-ED$_1$. The best results are obtained without setting any limit $R_{max}$. For both cases, $l = 90$ (after downsampling by 100) s.t. the time-series for first run is used for learning the LSTM-ED model.

### 6.3.2 Results and observations
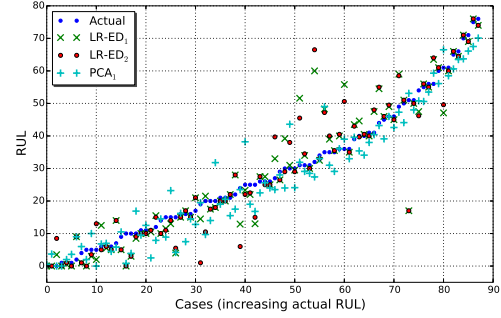


(a) Material-1
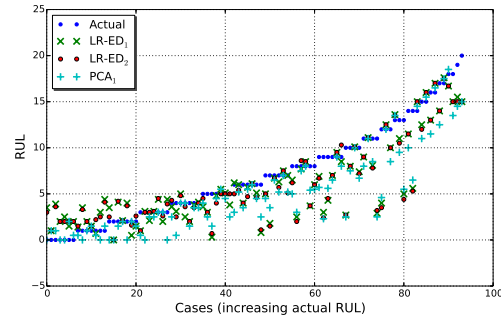


(b) Material-2

**Figure 8: Milling Machine: RUL predictions at each cycle after interpolation. Note: For material-1, estimates for every 5th cycle of all cases are shown for clarity.**

| | Material-1 | | | Material-2 | | |
|---|---|---|---|---|---|---|
| Metric | $PCA_1$ | $LR\text{-}ED_1$ | $LR\text{-}ED_2$ | $PCA_1$ | $LR\text{-}ED_1$ | $LR\text{-}ED_2$ |
| MAE | **4.2** | 4.9 | 4.8 | 2.1 | **1.7** | 1.8 |
| MSE | **29.9** | 71.4 | 65.7 | 8.2 | **7.1** | 7.8 |
| $MAPE_1(\%)$ | **25.4** | 26.0 | 28.4 | 35.7 | **31.7** | 35.5 |
| $MAPE_2(\%)$ | **9.2** | 10.6 | 10.8 | 14.7 | **11.6** | 12.6 |

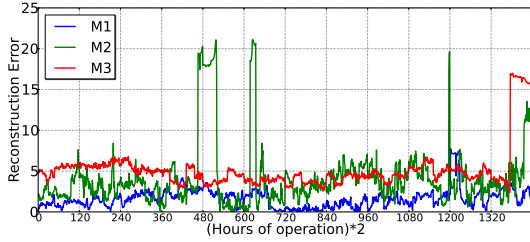**Table 2: Milling Machine: Performance Comparison**



**Figure 9: Pulverizer Mill: Pointwise reconstruction errors for last 30 days before maintenance.**

Figs. 7(a) and 7(e) show the variation of average reconstruction error from LSTM-ED w.r.t. the fraction of life passed for both materials. As shown, this error increases with amount of life passed, and hence is an appropriate indicator of health.

Figs. 7 and 8 show results based on every cycle of the data after interpolation, except when explicitly mentioned in the figure. The performance metrics on the original data points in the data set are summarized in Table 2. We observe that the first PCA component ($PCA_1$, $p = 1$) gives better results than LR-Lin and LR-Exp models with $p \geq 2$, and hence we present results for $PCA_1$ in Table 2. It is to be noted that for $p = 1$, all the four models LR-Lin, LR-Exp, $LR\text{-}ED_1$, and $LR\text{-}ED_2$ will give same results since all models will predict a different linearly scaled value of the first PCA component. $PCA_1$ and $LR\text{-}ED_1$ are the best models for material-1 and material-2, respectively. We observe that our best models perform well as depicted in histograms in Fig. 7. For the last few cycles, when actual RUL is low, an error of even 1 in RUL estimation leads to $MAPE_1$ of 100%. Figs. 7(b-d), 7(f-h) show the error distributions for different models for the two materials. As can be noted, most of the RUL prediction errors (around 70%) lie in the ranges [-4, 6] and [-3, 1] for material types 1 and 2, respectively. Also, Figs. 8(a) and 8(b) show predicted and actual RULs for different models for the two materials.

## 6.4 Pulverizer Mill Dataset

This dataset consists of readings for 6 sensors (such as bearing vibration, feeder speed, etc.) for over three years of operation of a pulverizer mill. The data corresponds to sensor readings taken every half hour between four consecutive scheduled maintenances $M_0$, $M_1$, $M_2$, and $M_3$, s.t. the operational period between any two maintenances is roughly one year. Each day's multivariate time-series data with length $l = 48$ is considered to be one subsequence. Apart from these scheduled maintenances, maintenances are done in between whenever the mill develops any unexpected fault affecting its normal operation. The costs incurred for any of the scheduled maintenances and unexpected maintenances are available. We consider the (z-normalized) original sensor readings directly for analysis rather than computing the PCA based derived sensors.

We assume the mill to be healthy for the first 10% of

| Maint. ID | $t_E$ | $P(E > t_E)$ | $t_C$ | $P(C > t_C \mid E > t_E)$ | E (last day) | $C(M_i)$ |
|---|---|---|---|---|---|---|
| $M_1$ | 1.50 | 0.25 | 7 | 0.61 | 2.4 | 92 |
| $M_2$ | 1.50 | 0.57 | 7 | 0.84 | 8.0 | 279 |
| $M_3$ | 1.50 | 0.43 | 7 | 0.75 | 16.2 | 209 |

**Table 3: Pulverizer Mill: Correlation between reconstruction error $E$ and maintenance cost $C(M_i)$.**

the days of a year between any two consecutive time-based maintenances $M_i$ and $M_{i+1}$, and use the corresponding subsequences for learning LSTM-ED models. This data is divided into training and validation sets. A different LSTM-ED model is learnt after each maintenance. The architecture with minimum average reconstruction error over a validation set is chosen as the best model. The best models learnt using data after $M_0$, $M_1$ and $M_2$ are obtained for $c = 40$, 20, and 100, respectively. The LSTM-ED based reconstruction error for each day is z-normalized using the mean and standard deviation of the reconstruction errors over the sequences in validation set.

From the results in Table 3, we observe that average reconstruction error $E$ on the last day before $M_1$ is the least, and so is the cost $C(M_1)$ incurred during $M_1$. For $M_2$ and $M_3$, $E$ as well as corresponding $C(M_i)$ are higher compared to those of $M_1$. Further, we observe that for the days when average value of reconstruction error $E > t_E$, a large fraction (>0.61) of them have a high ongoing maintenance cost $C > t_C$. The significant correlation between reconstruction error and cost incurred suggests that the LSTM-ED based reconstruction error is able to capture the health of the mill.

## 7. RELATED WORK

Similar to our idea of using reconstruction error for modeling normal behavior, variants of Bayesian Networks have been used to model the joint probability distribution over the sensor variables, and then using the evidence probability based on sensor readings as an indicator of normal behaviour (e.g. [11, 35]). [25] presents an unsupervised approach which does not assume a form for the HI curve and uses discrete Bayesian Filter to recursively estimate the health index values, and then use the k-NN classifier to find the most similar offline models. Gaussian Process Regression (GPR) is used to extrapolate the HI curve if the best class-probability is lesser than a threshold.

Recent review of statistical methods is available in [31, 36, 15]. We use HI Trajectory Similarity Based Prediction (TSBP) for RUL estimation similar to [40] with the key difference being that the model proposed in [40] relies on exponential assumption to learn a regression model for HI estimation, whereas our approach uses LSTM-ED based HI to learn the regression model. Similarly, [29] proposes RULCLIPPER (RC) which to the best of our knowledge has shown the best performance on C-MAPSS when compared to various approaches evaluated using the dataset [31]. RC tries various combinations of features to select the best set of features and learns linear regression (LR) model over these features to predict health index which is mapped to a HI polygon, and similarity of HI polygons is used instead of univariate curve matching. The parameters of LR model are estimated by assuming exponential target HI (refer Eq. 2). Another variant of TSBP [39] directly uses multiple PCA sensors for multi-dimensional curve matching which is used for RUL estimation, whereas we obtain a univariate HI by taking a weighted combination of the PCA sensors learnt

using the unsupervised HI obtained from LSTM-ED as the target HI. Similarly, [20, 19] learn a composite health index based on the exponential assumption.

Many variants of neural networks have been proposed for prognostics (e.g. [3, 27, 13, 18]): very recently, deep Convolutional Neural Networks have been proposed in [3] and shown to outperform regression methods based on Multi-Layer Perceptrons, Support Vector Regression and Relevance Vector Regression to directly estimate the RUL (see Table 4 for performance comparison with our approach on the Turbofan Engine dataset). The approach uses deep CNNs to learn higher level abstract features from raw sensor values which are then used to directly estimate the RUL, where RUL is assumed to be constant for a fixed number of starting cycles and then assumed to decrease linearly. Similarly, Recurrent Neural Network has been used to directly estimate the RUL [13] from the time-series of sensor readings. Whereas these models directly estimate the RUL, our approach first estimates health index and then uses curve matching to estimate RUL. To the best of our knowledge, our LSTM Encoder-Decoder based approach performs significantly better than Echo State Network [27] and deep CNN based approaches [3] (refer Table 4).

Reconstruction models based on denoising autoencoders [23] have been proposed for novelty/anomaly detection but have not been evaluated for RUL estimation task. LSTM networks have been used for anomaly/fault detection [22, 7, 41], where deep LSTM networks are used to learn prediction models for the normal time-series. The likelihood of the prediction error is then used as a measure of anomaly. These models predict time-series in the future and then use prediction errors to estimate health or novelty of a point. Such models rely on the the assumption that the normal time-series should be predictable, whereas LSTM-ED learns a representation from the entire sequence which is then used to reconstruct the sequence, and therefore does not depend on the predictability assumption for normal time-series.

# 8. DISCUSSION

We have proposed an unsupervised approach to estimate health index (HI) of a system from multi-sensor time-series data. The approach uses time-series instances corresponding to healthy behavior of the system to learn a reconstruction model based on LSTM Encoder-Decoder (LSTM-ED). We then show how the unsupervised HI can be used for estimating remaining useful life instead of relying on domain knowledge based degradation models. The proposed approach shows promising results overall, and in some cases performs better than models which rely on assumptions about health degradation for the Turbofan Engine and Milling Machine datasets. Whereas models relying on assumptions such as exponential health degradation cannot attune themselves to instance specific behavior, the LSTM-ED based HI uses the temporal history of sensor readings to predict the HI at a point. A case study on real-world industry dataset of a pulverizer mill shows signs of correlation between LSTM-ED based reconstruction error and the cost incurred for maintaining the mill, suggesting that LSTM-ED is able to capture the severity of fault.

# 9. REFERENCES

[1] A. Agogino and K. Goebel. Milling data set. *BEST lab, UC Berkeley*, 2007.

[2] G. Anand, A. H. Kazmi, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff. Deep temporal features to predict repeat buyers. In *NIPS 2015 Workshop: Machine Learning for eCommerce*, 2015.

[3] G. S. Babu, P. Zhao, and X.-L. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Database Systems for Advanced Applications*. Springer, 2016.

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[5] F. Cadini, E. Zio, and D. Avram. Model-based monte carlo state estimation for condition-based component replacement. *Reliability Engg. & System Safety*, 2009.

[6] F. Camci, O. F. Eker, S. Başkan, and S. Konur. 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, 2016.

[7] S. Chauhan and L. Vig. Anomaly detection in ecg time signals via deep long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–7. IEEE, 2015.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[9] J. B. Coble. Merging data sources to predict remaining useful life–an automated method to identify prognostic parameters. 2010.

[10] C. Croarkin and P. Tobias. Nist/sematech e-handbook of statistical methods. *NIST/SEMATECH*, 2006.

[11] A. Dubrawski, M. Baysek, M. S. Mikus, C. McDaniel, B. Mowry, L. Moyer, J. Ostlund, N. Sondheimer, and T. Stewart. Applying outbreak detection algorithms to prognostics. In *AAAI Fall Symposium on Artificial Intelligence for Prognostics*, 2007.

[12] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

[13] F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In *International Conference on Prognostics and Health Management*, 2008.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] M. Jouin, R. Gouriveau, D. Hissel, M.-C. Péra, and N. Zerhouni. Particle filter-based prognostics: Review, discussion and perspectives. *Mechanical Systems and Signal Processing*, 2015.

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

[17] J. Lam, S. Sankararaman, and B. Stewart. Enhanced trajectory based similarity prediction with uncertainty quantification. *PHM 2014*, 2014.

[18] J. Liu, A. Saxena, K. Goebel, B. Saha, and W. Wang. An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. Technical report, DTIC Document, 2010.

[19] K. Liu, A. Chehade, and C. Song. Optimize the signal quality of the composite health index via data fusion for degradation modeling and prognostic analysis. 2015.

[20] K. Liu, N. Z. Gebraeel, and J. Shi. A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. *Automation Science and Engineering, IEEE Transactions on*, 10(3):652–664, 2013.

[21] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *33rd International Conference on Machine Learning, Anomaly Detection Workshop. arXiv preprint arXiv:1607.00148*, 2016.

[22] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. 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*, 2015.

[23] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

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

[25] A. Mosallam, K. Medjaher, and N. Zerhouni. Component based data-driven prognostics for complex systems: Methodology and applications. In *Reliability Systems Engineering (ICRSE), 2015 First International Conference on*, pages 1–7. IEEE, 2015.

[26] E. Myötyri, U. Pulkkinen, and K. Simola. Application of stochastic filtering for lifetime prediction. *Reliability Engineering & System Safety*, 91(2):200–208, 2006.

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

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

[29] E. Ramasso. Investigating computational geometry for failure prognostics in presence of imprecise health indicator: Results and comparisons on c-mapss datasets. In *2nd Europen Confernce of the Prognostics and Health Management Society.*, 2014.

[30] E. Ramasso, M. Rombaut, and N. Zerhouni. Joint prediction of continuous and discrete states in time-series based on belief functions. *Cybernetics, IEEE Transactions on*, 43(1):37–50, 2013.

[31] E. Ramasso and A. Saxena. Performance benchmarking and analysis of prognostic methods for cmapss datasets. *Int. J. Progn. Health Manag*, 2014.

[32] A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher. Metrics for evaluating performance of prognostic techniques. In *Prognostics and health management, 2008. phm 2008. international conference on*, pages 1–17. IEEE, 2008.

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

[34] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*. 2014.

[35] D. Tobon-Mejia, K. Medjaher, and N. Zerhouni. Cnc machine tool's wear diagnostic and prognostic by using dynamic bayesian networks. *Mechanical Systems and Signal Processing*, 28:167–182, 2012.

[36] K. L. Tsui, N. Chen, Q. Zhou, Y. Hai, and W. Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015, 2015.

[37] P. Wang and G. Vachtsevanos. Fault prognostics using dynamic wavelet neural networks. *AI EDAM*, 2001.

[38] P. Wang, B. D. Youn, and C. Hu. A generic probabilistic framework for structural health prognostics and uncertainty management. *Mechanical Systems and Signal Processing*, 28:622–637, 2012.

[39] T. Wang. *Trajectory similarity based prediction for remaining useful life estimation*. PhD thesis, University of Cincinnati, 2010.

[40] T. Wang, J. Yu, D. Siegel, and J. Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *Prognostics and Health Management, International Conference on*. IEEE, 2008.

[41] M. Yadav, P. Malhotra, L. Vig, K. Sriram, and G. Shroff. Ode-augmented training improves anomaly detection in sensor data from machines. In *NIPS Time Series Workshop. arXiv preprint arXiv:1605.01534*, 2015.

[42] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

# APPENDIX

## A. BENCHMARKS ON TURBOFAN ENGINE DATASET

| Approach | S | A | MAE | MSE | MAPE$_1$ | MAPE$_2$ | FPR | FNR |
|---|---|---|---|---|---|---|---|---|
| **LR-ED$_2$ (proposed)** | 256 | **67** | **9.9** | **164** | 18 | **5.0** | **13** | 20 |
| **RULCLIPPER [29]** | **216** | **67** | 10.0 | 176 | 20 | NR | 56 | 44 |
| Bayesian-1 [24] | NR | NR | NR | NR | 12 | NR | NR | NR |
| Bayesian-2 [25] | NR | NR | NR | NR | **11** | NR | NR | NR |
| HI-SNR [19] | NR | NR | NR | NR | NR | 8 | NR | NR |
| ESN-KF [27] | NR | NR | NR | 4026 | NR | NR | NR | NR |
| EV-KNN [30] | NR | 53 | NR | NR | NR | NR | 36 | **11** |
| IBL [16] | NR | 54 | NR | NR | NR | NR | 18 | 28 |
| Shapelet [16] | 652 | NR | NR | NR | NR | NR | NR | NR |
| DeepCNN [3] | 1287 | NR | NR | 340 | NR | NR | NR | NR |

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