Data-driven Prognostics with Predictive Uncertainty Estimation using Ensemble of Deep Ordinal Regression Models

Vishnu TV, Diksha, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff

TCS Research, New Delhi, India
{*vishnu.tv, diksha.7,malhotra.pankaj,lovekesh.vig, gautam.shroff*}*@tcs.com*

Abstract

Prognostics or Remaining Useful Life (RUL) Estimation from multi-sensor time series data is useful to enable condition-based maintenance and ensure high operational availability of equipment. We propose a novel deep learning based approach for Prognostics with Uncertainty Quantification that is useful in scenarios where: (i) access to labeled failure data is scarce due to rarity of failures (ii) inherent noise is present in the sensor readings. The two scenarios mentioned are unavoidable sources of uncertainty in the RUL estimation process, often resulting in unreliable RUL estimates. To address (i), we formulate RUL estimation as an Ordinal Regression (OR) problem and propose LSTM-OR: deep Long Short Term Memory (LSTM) network-based approach to learn the OR function. We show that LSTM-OR naturally allows for the incorporation of censored operational instances in training along with the failed instances, leading to more robust learning. To address (ii), we propose a simple yet effective approach to quantify predictive uncertainty in the RUL estimation models by training an ensemble of LSTM-OR models. Through empirical evaluation on the publicly available turbofan engine benchmark datasets, we demonstrate that LSTM-OR is at par with commonly used deep metric regression-based approaches for RUL estimation when sufficient failed instances are available for training. Importantly, LSTM-OR outperforms these metric regression-based approaches in the practical scenario where failed training instances are scarce, but sufficient operational (censored) instances are additionally available. Furthermore, our uncertainty quantification approach yields high-quality predictive uncertainty estimates while also leading to improved RUL estimates compared to single best LSTM-OR models.

1. Introduction

In the current digital era, streaming data is ubiquitous. In the context of the Industrial Internet of Things, remote health monitoring services driven by sensor-driven data analytics are becoming increasingly popular. Data-driven approaches for anomaly detection, diagnostics, prognostics, and optimization have been proposed to provide operational support to engineers, ensure high reliability and availability of equipment, and to optimize the operational cost (Da Xu, He, & Li, 2014). Typically, a large number of sensors (order of hundreds or sometimes thousands) are installed to capture the operational behavior of sophisticated equipment with various sub-systems interacting with each other.

Recently, deep learning approaches have been proposed for various data-driven health monitoring tasks including anomaly detection (Malhotra, Vig, Shroff, & Agarwal, 2015; Malhotra, Ramakrishnan, et al., 2016; Gugulothu, Malhotra, Vig, & Shroff, 2018) and prognostics (Malhotra, TV, et al., 2016; Gugulothu et al., 2017; Zheng, Ristovski, Farahat, & Gupta, 2017), yielding state-of-the-art results for RUL estimation (Gugulothu et al., 2017) using Recurrent Neural Networks (RNNs). In this work, we focus on the problem of prognostics or Remaining Useful Life (RUL) estimation of operational instances given the current and historical readings from various sensors capturing their behavior. Deep learning approaches for prognostics, and equipment health monitoring in general, have certain limitations as they require large amount of labeled training data, the outcomes and decisions are difficult to interpret, etc. as highlighted in (Gugulothu, TV, et al., 2018; Gugulothu et al., 2017; Khan & Yairi, 2018).

In this work, we address two crucial practical challenges in deep learning based RUL estimation approaches. The challenges addressed and the corresponding key contributions of this work are as follows:

Challenge-I: Deep neural networks are prone to overfitting and typically require a large number of labeled training instances to avoid overfitting. If failure time for an instance is known, a target RUL can be obtained at any time before the failure time. However, labeled training instances for RUL estimation are few as failures are rare. Also, any operational instance (or any instance for which failure time is not known, or which has not failed yet) is considered to be *censored* as

Vishnu TV et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

target RUL cannot be determined for such an instance.

We note that deep RNNs (Heimes, 2008; Malhotra, TV, et al., 2016; Gugulothu et al., 2017; Zheng et al., 2017; Y. Zhang, Xiong, He, & Pecht, 2018) and Convolutional Neural Networks (CNNs) (Babu, Zhao, & Li, 2016) based approaches formulate RUL estimation as a metric regression (MR) problem where a normalized estimate of RUL is obtained given time series of sensor data via a non-linear regression metric function learned from the data. This MR formulation of RUL estimation cannot directly leverage censored data typically encountered in RUL estimation scenarios.

Key Contribution-I : In addition to using failed instances for training, we propose a novel approach to **leverage the censored instances** in a supervised learning setting, in turn, increasing the training data and leading to more robust RUL estimation models. We cast RUL estimation as an **ordinal regression** (Harrell, 2001; Orozco, Abbati, & Roberts, 2018) problem (instead of the typically used metric regression formulation) and propose LSTM-OR (Long Short Term Memory Networks based Ordinal Regression) based RUL Estimation approach. We show that *partially labeled training instances* can be generated from the readily available operational (non-failed) instances to augment the labeled training data in the ordinal regression setting to build a more robust RUL estimation models. We empirically show that LSTM-OR outperforms LSTM-MR by effectively leveraging censored data when the number of failed instances available for training is small.

Challenge-II: The black-box nature of deep neural networks makes it challenging to interpret the predictions/estimates, and in turn, gauge the reliability of the predictions. It is, therefore, desirable to **quantify the predictive uncertainty** in deep neural network based predictions of RUL - it can aid engineers and operators in risk assessment and decision making while accounting for the reliability of predictions.

Key Contribution-II: We propose a simple yet effective approach to **quantify uncertainty based on an ensemble of LSTM-OR models** (using similar idea as in (Lakshminarayanan, Pritzel, & Blundell, 2017) as detailed in Section 5). The ensemble of deep LSTM-OR models leads to improved RUL estimation performance, and also, the empirical standard deviation (ESD) of the predictions from LSTM-OR models provides an approximate measure of uncertainty. We empirically show that when ESD (i.e., the uncertainty in estimation) is low, the corresponding error in estimation is also low, making ESD a useful uncertainty quantification metric.

Organization of the paper: We provide an overview of related literature in Section 2. In Section 3, we briefly introduce deep LSTM networks as used to build our deep OR models. We provide details of LSTM-OR and uncertainty quantification approaches in Sections 4 and 5, respectively. We provide experimental evaluation details and observations in Section 6, and finally conclude in Section 7. Further, we provide an Appendix section for the detailed analysis of our uncertainty quantification approach.

2. RELATED WORK

Trajectory Similarity based RUL estimation: An important class of approaches for RUL estimation is based on trajectory similarity, e.g. (Wang, Yu, Siegel, & Lee, 2008; Khelif, Malinowski, Chebel-Morello, & Zerhouni, 2014; Lam, Sankararaman, & Stewart, 2014; Malhotra, TV, et al., 2016; Gugulothu et al., 2017). These approaches compare the health index trajectory or trend of a test instance with the trajectories of failed train instances to estimate RUL using a distance metric such as Euclidean distance. Such approaches work well when trajectories are smooth and monotonic in nature but are likely to fail in scenarios when there is noise or intermittent disturbances (e.g., spikes, operating mode change, etc.) as the distance metric may not be robust to such scenarios (Gugulothu et al., 2017).

Metric Regression based RUL estimation: Another class of approaches is based on metric regression. Unlike trajectory similarity based methods which rely on comparison of trends, metric regression methods attempt to learn a function to map sensor data to RUL directly, e.g., (Heimes, 2008; Benkedjouh, Medjaher, Zerhouni, & Rechak, 2013; Dong, Jin, Lou, & Wang, 2014; Babu et al., 2016; Gugulothu et al., 2017; Zheng et al., 2017; TV, Gupta, Malhotra, Vig, & Shroff, 2018). Such methods can better deal with non-monotonic and noisy scenarios by learning to focus on the relevant underlying trends irrespective of noise. Within metric regression methods, few methods consider non-temporal models such as Support Vector Regression for learning the mapping from values of sensors at a given time instance to RUL, e.g., (Benkedjouh et al., 2013; Dong et al., 2014).

Temporal models for RUL estimation: Deep temporal models such as those based on RNNs (Heimes, 2008; Malhotra, TV, et al., 2016; Gugulothu et al., 2017; Zheng et al., 2017) or Convolutional Neural Networks (CNNs) (Babu et al., 2016) can capture the degradation trends better compared to non-temporal models, and are proven to perform better. Despite the advantages of deep models, they are prone to overfitting in practical scenarios where the number of failed instances is small. Further, a set of techniques like (Ellefsen, Bjørlykhaug, Æsøy, Ushakov, & Zhang, 2019) uses semi-supervised learning for RUL estimation and proves to be advantageous in case of reduced amount of labeled training data. Our approach based on ordinal regression provisions for dealing with such scenarios by using censored instances in addition to failed instances to obtain more robust models. Our proposed approach is comparable to such approaches as we are using the cesored operational instances along with the failed instances in the training procedure.

Ordinal Regression for Survival Analysis: Ordinal Regression has been extensively used for applications such as age estimation from facial images (Chang, Chen, & Hung, 2011; Yang, Lin, Chang, & Chen, 2013; Niu, Zhou, Wang, Gao, & Hua, 2016; H. Liu, Lu, Feng, & Zhou, 2017), however; the applications are restricted to non-temporal image data using Convolutional Neural Networks. (Cheng, Wang, & Pollastri, 2008; Luck, Sylvain, Cardinal, Lodi, & Bengio, 2017) use feed-forward neural networks based ordinal regression for survival analysis. To the best of our knowledge, the proposed LSTM-OR approach is the first attempt to leverage ordinal regression based training using temporal LSTM networks for RUL estimation.

Deep Survival Analysis: A set of techniques for deep survival analysis have been proposed in the medical domain, e.g. (Katzman et al., 2018; Luck et al., 2017). However, it is not clear as to how such approaches can be adapted for RUL estimation applications, as they focus on estimating the survival probability at a given point in time, and cannot provide RUL estimates. On the other hand, LSTM-OR is capable of providing RUL estimates using time series sensor data.

Uncertainty quantification in RUL estimation models: Uncertainty analysis in data-driven equipment health monitoring is an active area of research and an unsolved problem. The approaches described in (Sankararaman & Goebel, 2013), (Sankararaman, Daigle, Saxena, & Goebel, 2013) use analytical algorithms to estimate the uncertainty in prognostics. They consider various sources of uncertainty, such as the loading and operating conditions of the system at hand, inaccurate sensor measurements, etc. to quantify their combined effect on RUL predictions. The various types of uncertainty are propagated through state space models until failure. Also, the future states of the system are estimated using the state space models.

Unlike these approaches, we focus on estimating RUL as well as predictive uncertainty by using an ensemble of deep neural networks to model the time-series of sensor data available till a given point in time, without predicting the future states of the system. Further, domain knowledge of the underlying dynamics of a system is not needed to quantify uncertainty, and therefore, our approach is much simpler to adapt.

Uncertainty quantification for deep neural networks: Recently, (Gal & Ghahramani, 2016) proposed the use of dropout at the inference time to provide Bayesian approximation in the RUL estimation. Further, (Lakshminarayanan et al., 2017) proposed the use of an ensemble of neural networks for predictive uncertainty estimation and demonstrated their use in comparison to Bayesian methods. Similarly, we also use an ensemble of LSTM networks to estimate the empirical uncertainty in RUL predictions.

3. BACKGROUND: DEEP LSTM NETWORKS

We use a variant of LSTMs (Hochreiter & Schmidhuber, 1997) as described in (Zaremba, Sutskever, & Vinyals, 2014) in the hidden layers of the neural network. Hereafter, we denote column vectors by bold small letters and matrices by bold capital letters. For a hidden layer with h LSTM units, the values for the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, output gate $\mathbf{o}_t$, hidden state $\mathbf{z}_t$, and cell state $\mathbf{c}_t$ at time t are computed using the current input $\mathbf{x}_t$, the previous hidden state $\mathbf{z}_{t-1}$, and the cell state $\mathbf{c}_{t-1}$, where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{z}_t$, and $\mathbf{c}_t$ are real-valued h-dimensional vectors.

Consider $W_{n_1,n_2} : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$ to be an affine transform of the form $\mathbf{z} \mapsto \mathbf{W}\mathbf{z} + \mathbf{b}$ for matrix $\mathbf{W}$ and vector $\mathbf{b}$ of appropriate dimensions. In the case of a multi-layered LSTM network with L layers and h units in each layer, the hidden state $\mathbf{z}_t^l$ at time t for the l-th hidden layer is obtained from the hidden state at $t-1$ for that layer $\mathbf{z}_{t-1}^l$ and the hidden state at t for the previous $(l-1)$-th hidden layer $\mathbf{z}_t^{l-1}$. The time series goes through the following transformations iteratively at l-th hidden layer for $t = 1$ through T, where T is the length of the time series:

$$\begin{pmatrix} \mathbf{i}_t^l \\ \mathbf{f}_t^l \\ \mathbf{o}_t^l \\ \mathbf{g}_t^l \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W_{2h,4h} \begin{pmatrix} \mathbf{D}(\mathbf{z}_t^{l-1}) \\ \mathbf{z}_{t-1}^l \end{pmatrix} \tag{1}$$

where the cell state $\mathbf{c}_t^l$ is given by $\mathbf{c}_t^l = \mathbf{f}_t^l \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \mathbf{g}_t^l$, and the hidden state $\mathbf{z}_t^l$ is given by $\mathbf{z}_t^l = \mathbf{o}_t^l tanh(\mathbf{c}_t^l)$. We use dropout for regularization (Pham, Bluche, Kermorvant, & Louradour, 2014), which is applied only to the non-recurrent connections, ensuring information flow across time-steps for any LSTM unit. The dropout operator $\mathbf{D}(\cdot)$ randomly sets the dimensions of its argument to zero with probability equal to a dropout rate. The sigmoid (σ) and $tanh$ activation functions are applied element-wise.

In a nutshell, this series of transformations for $t = 1 \dots T$, converts the input time series $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_T$ of length T to a fixed-dimensional vector $\mathbf{z}_T^L \in \mathbb{R}^h$. We, therefore, represent the LSTM network by a function f_{LSTM} such that $\mathbf{z}_T^L = f_{LSTM}(\mathbf{x}; \mathbf{W})$, where $\mathbf{W}$ represents all the parameters of the LSTM network.

4. DEEP ORDINAL REGRESSION FOR RUL ESTIMATION

4.1. Terminology

Consider a learning set $\mathcal{D} = \{\mathbf{x}^i, r^i\}_{i=1}^n$ of n failed instances, where r^i is the target RUL, $\mathbf{x}^i = \mathbf{x}_1^i \dots \mathbf{x}_{T^i}^i \in \mathcal{X}$ is a mul-

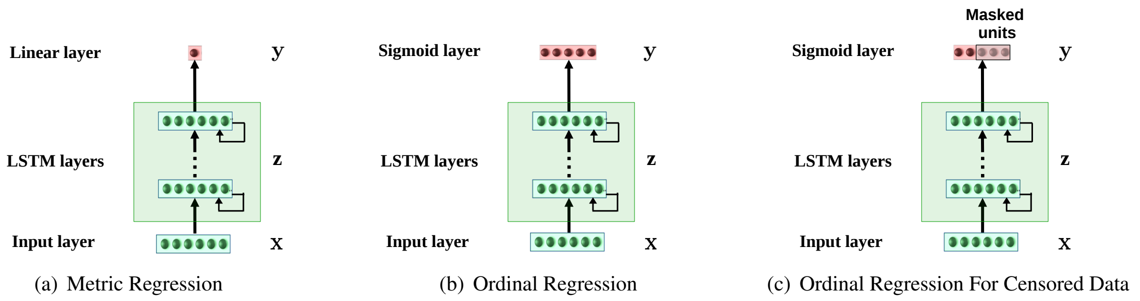
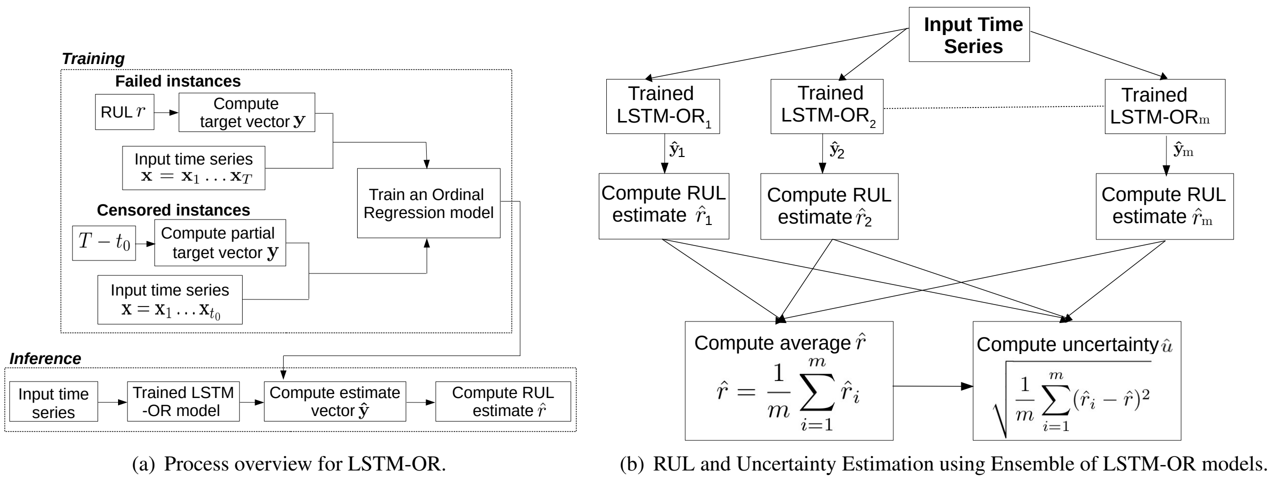


Figure 1. Deep Ordinal Regression versus Deep Metric Regression.



Training
Failed instances
RUL r
Compute target vector $\mathbf{y}$
Input time series $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_T$
Censored instances
$T - t_0$
Compute partial target vector $\mathbf{y}$
Input time series $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_{t_0}$
Train an Ordinal Regression model
Inference
Input time series
Trained LSTM -OR model
Compute estimate vector $\hat{\mathbf{y}}$
Compute RUL estimate $\hat{r}$
(a) Process overview for LSTM-OR.
Input Time Series
Trained LSTM-OR$_1$
Trained LSTM-OR$_2$
Trained LSTM-OR$_m$
$\hat{\mathbf{y}}_1$
$\hat{\mathbf{y}}_2$
$\hat{\mathbf{y}}_m$
Compute RUL estimate $\hat{r}_1$
Compute RUL estimate $\hat{r}_2$
Compute RUL estimate $\hat{r}_m$
Compute average $\hat{r}$
$\hat{r} = \frac{1}{m}\sum_{i=1}^{m}\hat{r}_i$
Compute uncertainty $\hat{u}$
$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r}_i - \hat{r})^2}$
(b) RUL and Uncertainty Estimation using Ensemble of LSTM-OR models.

Figure 2. Steps in LSTM-OR and Ensemble of LSTM-OR.

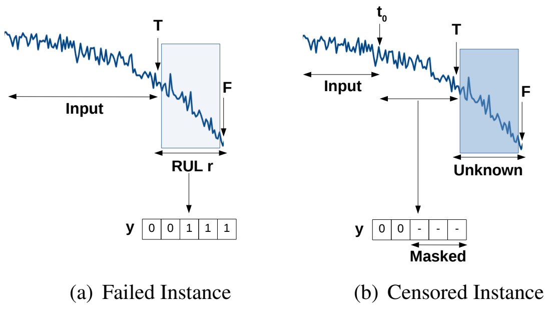


Figure 3. Target vector creation for failed versus censored instance.

tivariate time series of length T^i, $\mathbf{x}_t^i \in \mathbb{R}^p$, p is the number of input features (sensors). The total operational life of an instance i till the failure point is F^i, s.t. $T^i \leq F^i$. Therefore, $r^i = F^i - T^i$ is the RUL in a given unit of measurement, e.g., number of cycles or operational hours. Hereafter, we omit the superscript i in this section for better readability and provide all the formulation considering an instance (unless stated otherwise).

We consider an upper bound r_u on the possible values of target RUL for training as, in practice, it is not possible to predict too far ahead in the future. So if the target RUL $r > r_u$, we clip the value of r to r_u. The usually defined goal of RUL estimation via Metric Regression (MR) is to learn a mapping $f_{MR} : \mathcal{X} \rightarrow [0, r_u]$. With these definitions, we next describe LSTM-based Ordinal Regression (LSTM-OR) approach as summarized in Figure 2(a), and then describe how we incorporate censored data into the LSTM-OR formulation.

4.2. LSTM-based Ordinal Regression

Instead of mapping an input time series to a real-valued number as in MR, we break the range $[0, r_u]$ of RUL values into K intervals of length $c = \frac{r_u}{K}$ each, where each interval is then considered as a discrete variable. The j-th interval corresponds to $((j-1) \cdot c, j \cdot c)$, and r is mapped to the k-th interval with $k = \lceil \frac{r}{c} \rceil$, where $\lceil . \rceil$ denotes the ceiling function.

We consider K binary classification sub-problems for the K discrete variables (intervals): a classifier C_j solves the binary classification problem of determining whether $r \leq j\frac{r_u}{c}$. We train an LSTM network for the K binary classification tasks simultaneously by modeling them together as a multi-label classification problem: We obtain the multi-label target vector $\mathbf{y} = [y_1, \dots, y_K] \in \{0,1\}^K$ from r such that

$$y_j = \begin{cases} 0 & j < k \\ 1 & j \geq k \end{cases} \tag{2}$$

where $j = 1, 2, \dots, K$.

For example, consider a scenario where $K = 5$, and r maps to the third interval such that $k = 3$. The target is then given by $\mathbf{y} = [0, 0, 1, 1, 1]$, as illustrated in Figure 3(a). Effectively, the goal of LSTM-OR is to learn a mapping $f_{OR} : \mathcal{X} \rightarrow \{0,1\}^K$ by minimizing the loss function $\mathcal{L}_{OR}$ given by:

$$\begin{aligned} \mathbf{z}_T^L &= f_{LSTM}(\mathbf{x}; \mathbf{W}) \\ \hat{\mathbf{y}} &= \sigma(\mathbf{W}_C\, \mathbf{z}_T^L + \mathbf{b}_C) \\ \mathcal{L}_{OR}(\mathbf{y}, \hat{\mathbf{y}}) &= -\frac{1}{K}\sum_{j=1}^{K} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j) \end{aligned} \tag{3}$$

where, $\hat{\mathbf{y}}$ is the estimate for target $\mathbf{y}$, $\mathbf{W}$ represents the parameters of the LSTM network, and $\mathbf{W}_C$ and $\mathbf{b}_C$ are the parameters of the layer that maps $\mathbf{z}_T^L$ to the output sigmoid layer.

4.3. Using Censored Data for Training

For any censored instance, the data is available only till a time T prior to failure, and the failure time F is unknown (illustrated in Figure 3(b)). Therefore, the target RUL r is also unknown. However, at any time t_0 s.t. $1 \leq t_0 < T$, it is known that the RUL $r > T - t_0$ since the instance is operational at least till T. Considering $\mathbf{x} = \mathbf{x}_1 \dots \mathbf{x}_{t_0}$ as the input time series, we next show how we assign labels to few of the dimensions y_j of the target vector $\mathbf{y}$: Assuming $T - t_0$ maps to the interval $k' = \left\lceil \frac{T-t_0}{c} \right\rceil$, since $T - t_0 < r$, we have $\left\lceil \frac{T-t_0}{c} \right\rceil \leq \left\lceil \frac{r}{c} \right\rceil \implies k' \leq k$. Since k is unknown (as r is unknown) and we have $k' \leq k$, the target vector $\mathbf{y}$ can only be partially obtained:

$$y_j = \begin{cases} 0 & j < k' \\ unknown & j \geq k' \end{cases} \tag{4}$$

For all $j \geq k'$, the corresponding binary classifier targets are masked, as shown in Figure 3(b), and the outputs from these classifiers are not included in the loss function for the instance. The loss function $\mathcal{L}_{ORC}$ given by Equation 3 can thus be modified for including the censored instances in training as:

$$\mathcal{L}_{ORC}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{K'}\sum_{j=1}^{K'} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j) \tag{5}$$

where $K' = k' - 1$ for a censored instance and $K' = K$ for a failed instance.

Our approach can be seen as an instance of semi-supervised learning. We have labels for classes corresponding to the unmasked intervals while the labels for masked intervals are unknown. In our proposed approach, we are able to use the instances corresponding to the masked intervals despite incomplete label information. This approach is orthogonal to the other efforts made in semi-supervised based RUL approaches such as (Ellefsen et al., 2019) and can be combined together.

4.4. Mapping OR estimates to RUL

Once trained, each of the K classifiers provides a probability $\hat{y}_j$ for RUL being greater than the upper limit of the interval corresponding to the j-th classifier. We obtain the point-estimate $\hat{r}$ for r from $\hat{\mathbf{y}}$ for a test instance as follows (similar to (Chang et al., 2011)):

$$\hat{r} = r_u(1 - \frac{1}{K}\sum_{j=1}^{K} \hat{y}_j) \tag{6}$$

It is worth noting that once learned, the LSTM-OR model can be used in an online manner for operational instances: at current time instance t, the sensor data from the latest T time instances can be input to the model to obtain the RUL estimate r at t.

5. PREDICTIVE UNCERTAINTY QUANTIFICATION USING ENSEMBLE OF LSTM-OR MODELS

Uncertainty quantification is essential in the case of RUL estimation as equipment and operations involved are often critical, and reliable predictions close to (but of course, prior to) failures can help avoid catastrophic failures by generating suitable alarms beforehand. Lack of sufficient training data, inherent noise in sensor readings, and uncertainty in the future usage and operation of the equipment are few sources of uncertainty in the case of data-driven predictive models for RUL estimation.

Quantifying uncertainty in RUL estimates can assist ground engineers and operators to arrive at more informed decisions compared to scenarios where only RUL estimates are available without any metric indicating whether the model is particular about the estimate or not. In other words, uncertainty quantification of the RUL estimate enhances the reliability of data-driven models. This is even more relevant in deep neural network-based estimation models due to their otherwise black-box nature.

An uncertainty metric can be considered to be reliable if: i) for low uncertainty values, i.e. whenever the model is confident about its estimations, the corresponding errors in the RUL estimations are low, and for high uncertainty values, the corresponding errors in the RUL estimation model should be high, ii) it produces RUL estimates with low uncertainty when a failure is approaching, i.e. the model should be able to precisely estimate the RUL with a high degree of certainty close to failures.

To quantify the predictive uncertainty in the target vector estimate $\hat{\mathbf{y}}$ and the corresponding RUL estimate $\hat{r}$, we consider an ensemble learning approach. For training an ensemble of LSTM-OR models, we consider all the training data while using different (random) initializations of the parameters $(\mathbf{W}, \mathbf{W}_C, \mathbf{b}_C)$ of LSTM-OR models and random shuffling of the training instances to obtain m different models in an ensemble. We consider two measures for predictive uncertainty, i.e., Empirical Standard Deviation (ESD) and Entropy (ENT), defined as follows-

5.1. Empirical Standard Deviation (ESD)

The final RUL estimate of the ensemble is given by the simple average of the RUL estimates of the m models in the ensemble, and the empirical standard deviation (ESD) in the RUL estimates is used as an approximation of the predictive uncertainty in RUL estimation. More specifically, as shown in Figure 2(b), we train m LSTM-OR models such that we have m RUL estimates $\hat{r_i}$ for any instance, $i = 1, \ldots, m$. We obtain the point estimate $\hat{r}$ for r from $\hat{r_i}$ for an instance as follows:

$$\hat{r} = \frac{1}{m}\sum_{i=1}^{m}\hat{r}_i \tag{7}$$

The uncertainty $\hat{u}$ in terms of ESD is given by:

$$\hat{u}_{ESD} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r}_i - \hat{r})^2} \tag{8}$$

We use min-max normalization to normalize the uncertainty value ($\hat{u}_{ESD}$) for any instance where the minimum and maximum uncertainty values $\hat{u}_{ESD}$ are obtained from the instances in the hold-out validation set.

5.2. Entropy (ENT)

The uncertainty value in terms of entropy, similar to (Park & Simoff, 2015), can be defined as follows:

Let $\mathcal{Z}$ be the set of possible values of the multi-label target vector. In our case, since we are using OR, the number of possible combinations for any given K is $K+1$. For example, when $K = 3$, $\mathcal{Z} = \{[0,0,0],[0,0,1],[0,1,1],[1,1,1]\}$. For entropy, we average out the m estimates to get the final $\hat{\mathbf{y}} = \sum_{i=1}^{m}\hat{\mathbf{y}}_i$. The uncertainty for a test instance, considering the independent nature of all of the K binary classifier is given as follows:

$$\begin{gathered}\hat{u}_{ENT} = -\sum_{\mathbf{z}\in\mathcal{Z}} P(\mathbf{z})\, log(P(\mathbf{z}))\\ P(\mathbf{z}) = \prod_{j=1}^{K} P(z_j)\\ P(z_j) = \begin{cases}\hat{y}_j & z_j = 1\\ 1-\hat{y}_j & z_j = 0\end{cases}\end{gathered} \tag{9}$$

For example, consider the case with $m = 2$ and $K = 3$. Let $\hat{\mathbf{y}}_1$ be $[0.16, 0.82, 0.97]$ and $\hat{\mathbf{y}}_2$ be $[0.12, 0.86, 0.92]$, vectors corresponding to the m models of length K. The final $\hat{\mathbf{y}}$ vector is then given by the average of $\hat{\mathbf{y}_1}$ and $\hat{\mathbf{y}_2}$, $\hat{\mathbf{y}} = [0.14, 0.84, 0.945]$. Then, for each $\mathbf{z} \in \mathcal{Z}$, $P(\mathbf{z})$ is calculated as follows: suppose $\mathbf{z}$ is $[0,0,0]$, then $P(\mathbf{z})$ = $(0.86 \times 0.16 \times 0.055)$ = 0.008 and $P(\mathbf{z})log(P(\mathbf{z}))$ = -4.88. Similarly, when $\mathbf{z}$ is $[0,1,1]$, then $P(\mathbf{z})$ = $(0.86 \times 0.84 \times 0.945)$ = 0.68 and $P(\mathbf{z})log(P(\mathbf{z}))$ = -0.39. Finally, the uncertainty value $\hat{u}_{ENT}$ is given as the summation over such $K+1$ values, corresponding to each $\mathbf{z} \in \mathcal{Z}$.

We normalize the uncertainty values ($\hat{u}_{ENT}$) using the minimum and maximum uncertainty values across all instances in a hold-out validation set through min-max normalization. For entropy calculations, we also tried by calculating the entropy for each of the K classifiers individually and later averaging them out to get the final one. But, it was less effective as compared to the approach defined before.

We found ESD to be the most robust measure of uncertainty as compared to ENT. We support this with experimental evaluation in Section 6.3.

6. EXPERIMENTAL EVALUATION

We evaluate RUL estimation and uncertainty quantification approaches using the publicly available Commercial Modular Aero-propulsion System Simulation (C-MAPSS) aircraft turbofan engine benchmark datasets (Saxena & Goebel, 2008). We provide an overview of the dataset in Section 6.1. We consider metric regression models and ordinal regression models trained only on failed instances as baseline models, and compare following approaches for RUL estimation: i) **MR**: LSTM-MR using failed instances only (as in (Zheng et al., 2017; Heimes, 2008; Gugulothu et al., 2017)), ii) **OR**: LSTM-OR using failed instances only and using loss as in Equation 3, iii) **ORC**: LSTM-OR leveraging censored data along with failed instances using loss as in Equation 5. We describe RUL estimation approaches in Section 6.2 using

ORCE model, which is a simple average ensemble of ORC models. Further, to evaluate the uncertainty quantification approach as described in Section 5, we study the relationship of uncertainty estimates with error and ground truth RUL in Section 6.3 while also introducing novel metrics to evaluate the efficacy of uncertainty estimates in the context of prognostics.

6.1. Dataset Description

We consider datasets FD001 and FD004 from the simulated turbofan engine datasets[1] (Saxena & Goebel, 2008). The training sets (*train_FD001.txt* and *train_FD004.txt*) of the two datasets contain time series of readings for 24 sensors (21 sensors and 3 operating condition variables) of several instances (100 in FD001 and 249 in FD004) of a turbofan engine from the beginning of usage till the end of life. The time series for the instances in the test sets (*test_FD001.txt* and *test_FD004.txt*) are pruned sometime prior to failure, such that the instances are operational and their RUL needs to be estimated. The actual RUL values for the test instances are available in *RUL_FD001.txt* and *RUL_FD004.txt*. We randomly sample 20% of the available training set instances, as given in Table 1, to create a validation set for hyperparameter selection.

For simulating the scenario for censored instances, a percentage $p_c \in \{0, 50, 70, 90\}$ of the training and validation instances are randomly chosen, and time series for each instance is randomly truncated at one point prior to failure. We then consider these truncated instances as censored (currently operational) and their actual RUL values as unknown. The remaining $(100-p_c)\%$ of the instances are considered as failed. Further, the time series of each instance thus obtained (censored and failed) is truncated at 20 random points in the life prior to failure, and the exact RUL r for failed instances and the minimum possible RUL $T - t_0$ for the censored instances (as in Section 4 and Figure 3) at the truncated points are used for obtaining the models. The number of instances thus obtained for training and validation for $p_c = 0$ is given in Table 2. The test set remains the same as the benchmark dataset across all scenarios (with no censored instances). The MR and OR approaches cannot utilize the censored instances as the exact RUL targets are unknown, while ORC can utilize the lower bound on RUL targets to obtain partial labels as per Equation 4.

An engine may operate in different operating conditions and also have different failure modes at the end of its life. The number of operating conditions and failure modes for both the datasets is given in Table 1. FD001 has only one operating condition, so we ignore the corresponding three sensors, such that $p = 21$, whereas FD004 has six operating conditions determined by the three operating condition variables. We map these six operating conditions to a 6-dimensional one hot vector as in (Zheng et al., 2017), such that $p = 27$.

[1]`https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan`

6.2. RUL Estimation

In this section, we define performance metrics to evaluate our RUL estimation models, i.e. ORC and ORCE. Further, we discuss our experimental settings, which is followed by results and observations. We also draw a comparison between our proposed RUL estimation models and already existing RUL estimation models.

6.2.1. Performance Metrics for Evaluating RUL Estimation Models

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) and Root Mean Squared Error (RMSE): For a test instance i, the error in estimation is given by $e_i = \hat{r}_i - r_i$. The timeliness score for N test instances is given by $S = \sum_{i=1}^{N}(exp(\gamma \cdot |e_i|) - 1)$, where $\gamma = 1/\tau_1$ if $e_i < 0$, else $\gamma = 1/\tau_2$. Usually, $\tau_1 > \tau_2$ such that late predictions are penalized more compared to early predictions. We use $\tau_1 = 13$ and $\tau_2 = 10$ as proposed in (Saxena, Goebel, Simon, & Eklund, 2008). The lower the value of S, the better is the performance. The root mean squared error (RMSE) is given by: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}$.

6.2.2. Experimental Setup

Dataset	Train	Validation	Test	OC	FM
FD001	80	20	100	1	1
FD004	199	50	248	6	2

Table 1. Number of train, validation and test instances. Here, OC: number of operating conditions, FM: number of fault modes.

Dataset	Train	Validation	Test
FD001	1600	400	100
FD004	3980	1000	248

Table 2. Number of truncated instances.

We consider $r_u = 130$ cycles for all models, as used in (Babu et al., 2016; Zheng et al., 2017). For OR and ORC, we consider $K = 10$ such that interval length $c = 13$. For training the MR models, a normalized RUL in the range 0 to 1 (where 1 corresponds to a target RUL of 130 or more) is given as the target for each input. We use a maximum time series length of $T = 360$; for any instance with more than 360 cycles, we take the most recent 360 cycles. Also, we use standard z-normalization to normalize the multi-sensor input time series by using sensor-wise mean and standard deviation from the train set.

The hyperparameters h (number of hidden units per layer),

L (number of hidden layers) and the learning rate are chosen from the sets $\{50, 60, 70, 80, 90, 100\}$, $\{2, 3\}$ and $\{0.001, 0.005\}$, respectively. We use a dropout rate of 0.2 for regularization and a batch size of 32 during training. The models are trained for a maximum of 2000 iterations with early stopping. The best hyperparameters are obtained using a grid search by minimizing the respective loss function on the validation set.

6.2.3. Results and Observations

As summarized in Table 3, we observe that: As the number of failed training instances (n_f) decreases, the performance for all models degrades (as expected). However, importantly, for scenarios with small n_f, ORC significantly outperforms MR and OR. For example, as shown in Figure 4, with $p_c = 90\%$ (i.e. with $n_f = 8$ and 20 for FD001 and FD004, respectively), ORC performs significantly better than MR, and shows 15.1% and 3.6% improvement over MR in terms of RMSE, for FD001 and FD004, respectively. The gain in terms of timeliness score S for FD001 dataset is higher because of the exponential nature of S (refer Section 6.2.1). As S penalizes errors in RUL estimates exponentially, substantial gains with decreasing n_f in terms of S further prove the robustness of OR compared to MR.

While MR and OR have access to only a small number failed instances n_f for training, ORC has access to n_f failed instances as well as partial labels from n_c censored instances for training. Therefore, MR and OR models tend to overfit while ORC model is more robust. Further, the proposed ideas of ordinal regression and handling of censored data are orthogonal to the choice of the deep learning architecture, we hope that the same results would hold true for other existing approaches from literature such as those listed in Table 4 based on CNNs, DBNs and semi-supervised learning.

We further show the detailed performance comparison between ORC and ORCE in the Appendix A.1.

6.3. Uncertainty Quantification

For experiments related to uncertainty quantification, we consider ORCE, an ensemble of $m = 6$ models (we consider upto 10 models in the ensemble, and found $m = 6$ to work best across the scenarios considered). For selecting $m = 6$ models from available 10 models, we ordered the models in the ascending order of their respective loss values on the validation set and then selected the first 6 models. The models are trained on the best hyperparameters selected from the corresponding hyperparameter sets of ORC. While training different models, we ensure random initializations of the parameters of neural network and random shuffling of the training instances. Further, we introduce various metrics used to evaluate the performance of the proposed ensemble-based uncertainty estimation approach. Using these metrics, we demonstrate the efficacy of the proposed approach from a practical point of view. We compare the proposed ESD (Equation 8) and ENT (Equation9) for uncertainty evaluation.

6.3.1. Performance Metrics for Evaluating Uncertainty Quantification Methods

We expect our model to be precise (have high certainty) when the RUL estimates are correct, and less confident (have low certainty) for highly erroneous RUL estimates. We consider an RUL estimation to be correct if the absolute error $|r-\hat{r}| \leq \tau_e$, and to be certain if the corresponding uncertainty estimate $\hat{u} \leq \tau_u$. Also, for evaluating the performance of uncertainty metrics, we restrict the target RUL r to a maximum of $r_u = 130$ because we train our models with a maximum target RUL of r_u and so $\hat{r}$ cannot be higher than r_u. This is done because even if the model confidently estimates $\hat{r}$ close to r_u, a value of r much more significant than r_u will lead to high error and cannot result in proper performance evaluation of the uncertainty metrics. Under the above considerations, we measure precision and recall, refer Appendix A.2 for definitions, to evaluate the performance of the uncertainty quantification approach.

We analyze the relation of uncertainty with nearness to failure, to avoid fatal consequences. It is desirable to have low error as well as low uncertainty when r is low. To evaluate this aspect, we study the variation in precision for different RUL thresholds τ_r, considering test instances with low ground truth RULs. The modified precision P_l in this context is given by:

$$P_l = \frac{\#(r \leq \tau_r) \cap \#(\hat{u} \leq \tau_u) \cap \#(|r-\hat{r}| \leq \tau_e)}{\#(r \leq \tau_r) \cap \#(\hat{u} \leq \tau_u))} \tag{10}$$

For given thresholds τ_r and τ_u, P_l quantifies the fraction of test instances with actual RUL $r \leq \tau_r$ and uncertainty $\leq \tau_u$ that also have error $\leq \tau_e$.

6.3.2. Results and Observations

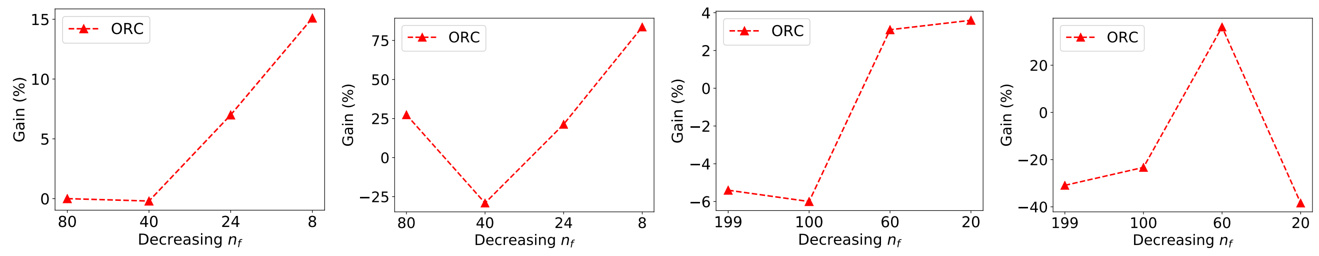
For the sake of brevity, we restrict the results and observations to the uncensored scenario, i.e., $p_c = 0\%$. Similar results and observations for models corresponding to censored scenarios are presented in Appendix A.3.

Comparing ESD vs Entropy (ENT) as uncertainty metric: Precision and Recall (as in Equation 11) are used to compare the two approaches for uncertainty estimation. Precision-Recall curves are obtained by varying the threshold on uncertainty $0.1 \leq \tau_u \leq 1.5$ while keeping $\tau_e = 10$. We observe that for $R \geq 0.1$, P is higher in case of ESD for FD001 dataset, shown in Figure 6(a). Similar behavior is observed in case of FD004 dataset, for $R \geq 0.2$, shown in Figure 6(b).

We further plot $F1$ score (as in Equation 11) by varying the

	FD001								**FD004**							
	Instances		RMSE			Timeliness Score (S)			Instances		RMSE			Timeliness Score (S) $\times 10^{-3}$		
$p_c(\%)$	n_f	n_c	MR	OR	ORC	MR	OR	ORC	n_f	n_c	MR	OR	ORC	MR	OR	ORC
0	80	0	**15.62**	15.63	15.63	507.2	**367.64**	**367.64**	199	0	**26.88**	28.33	28.33	**4.92**	6.44	6.44
50	40	40	**17.56**	19.06	17.60	**444.1**	564.14	572.63	100	99	**29.71**	32.85	31.48	**7.97**	17.9	9.83
70	24	56	19.92	**16.48**	18.53	713.31	**362.21**	561.11	60	139	33.17	33.65	**32.13**	18.8	17.4	**12.0**
90	8	72	25.32	24.83	**21.51**	1.26×10^4	3.07×10^4	$\mathbf{20.64 \times 10^2}$	20	179	41.23	43.88	**39.75**	**102.0**	111.0	141.13

Table 3. Comparison of various LSTM-based approaches, considered in the terms of RMSE and Timeliness Score (S) for FD001 and FD004 datasets. n_f and n_c denote number of failed and censored instances in training set, respectively. For small n_f, OR and ORC perform significantly better than MR.



(a) RMSE FD001 (b) Timeliness Score (S) FD001 (c) RMSE FD004 (d) Timeliness Score (S) FD004

Figure 4. %age gain of ORC over MR with decreasing number of failed instances (n_f) in training.

	FD001		**FD004**	
Method	RMSE	S $\times 10^{-2}$	RMSE	S$\times 10^{-3}$
LSTM-MR (Zheng et al., 2017)	16.14	3.38	28.17	5.55
MR (ours)	15.62	5.07	26.88	4.92
OR/ORC (proposed)	15.63	3.68	28.33	6.44
CNN-MR (Babu et al., 2016)	18.45	1.29	29.16	7.89
MODBNE (C. Zhang, Lim, Qin, & Tan, 2016)	15.04	3.34	28.66	6.56
CNN + FNN (Li, Ding, & Sun, 2018)	12.61	2.74	23.31	12.47
Semi-supervised setup (Ellefsen et al., 2019)	12.56	2.31	22.66	2.84

Table 4. Performance comparison of the proposed approach with existing state-of-the-art approaches on the full dataset in terms of RMSE and Timeliness Score (S).

τ_u, shown in Figure 6(c) and 6(d), which shows that ESD is a better uncertainty quantification metric compared to ENT. (We also analyze the instances for which ESD has unexpected behavior in terms of low uncertainty while having a high error in RUL estimate. The observations are given in Appendix A.3.)

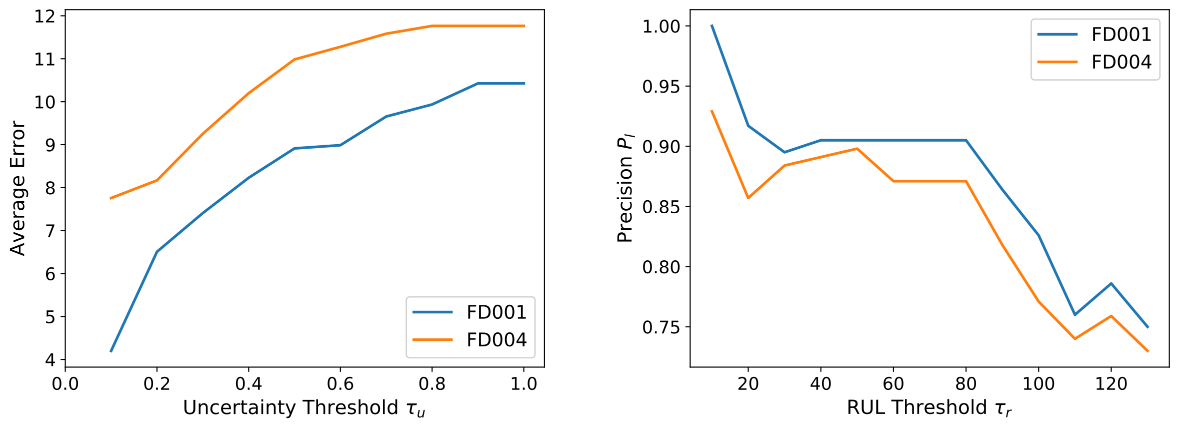
Relation between uncertainty and error: For a reliable model, RUL estimates with high certainty must be accurate, i.e., have low RUL estimation errors. To evaluate the performance of uncertainty metric in this context, we consider instances with uncertainty $\hat{u} \leq \tau_u$, and compute the average error in RUL estimation for these instances. As shown in Figure 5(a), we observe that for low values of τ_u, the average error thus computed is also low, indicating that the model is more accurate when it is more specific. Further, as expected, we observe an increase in average error with increasing τ_u, suggesting that the RUL estimates tend to be more erroneous when the model is uncertain. For censored data we have shown results in Figure 7, in Appendix.

Relation between uncertainty and actual RUL: For quantifying the relationship between RUL and uncertainty, P_l is calculated as in Equation 10. P_l is computed for varying τ_r, ranging from 10 to 130 and, keeping τ_u and τ_e fixed as 0.2 and 10 respectively. From a practical point of view, higher precision (P_l) in case of lower values of τ_r is expected to correctly and confidently handle instances that are approaching failure. A similar trend is observed in our case also, as shown in Figure 5(b). For $\tau_r = 20$, $P_l = 0.917$ for FD001 dataset and $P_l = 0.857$ for FD004 dataset suggests that the model is certain and accurate 91.7% of the times for FD001 dataset and 85.7% of the times for FD004 dataset. For censored data we have shown results in Figure 8, in Appendix.

7. CONCLUSION AND DISCUSSION

In this work, we have proposed a novel approach for RUL estimation using deep ordinal regression based on multilayered LSTM neural networks. We have argued that ordinal regression formulation is more robust compared to metric regression, as the former allows for the incorporation of more labeled data from censored instances. We found that leveraging censored instances significantly improves performance when the number of failed instances is small. In future, it would be interesting to see if a semi-supervised approach (e.g., as in (Yoon et al., 2017; Gugulothu, TV, et al., 2018)) with initial unsupervised pre-training of LSTMs using failed as well as censored instances can further improve the robustness of the models. Also, an extension to the proposed approach to address the usually encountered non-stationarity scenario using strategies similar to (Saurav et al., 2018) can be considered. It is to be noted that although we have experimented with LSTMs for Ordinal Regression, our OR approach is generic enough to be useful for any neural network, e.g. CNNs.

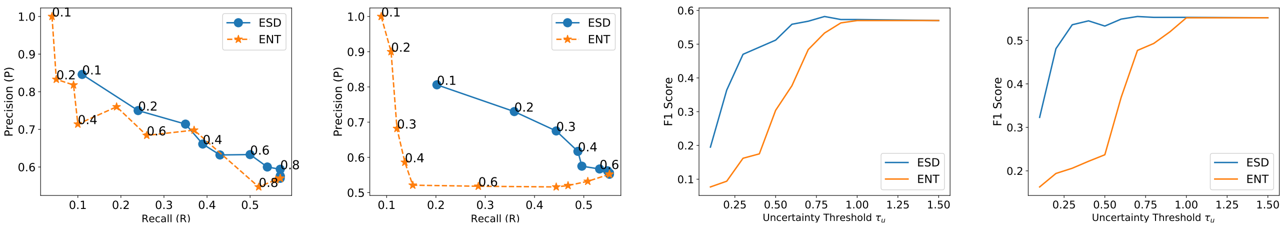
Further, we have proposed a simple yet effective approach to



(a) Average Error with varying uncertainty threshold.

(b) Uncertainty Evaluation with varying RUL.

Figure 5. Performance evaluation of ESD as an uncertainty metric showing: (a) lower uncertainty values corresponding to low RUL estimation errors, (b) highly precise and correct uncertainty estimates close to failures, i.e. when RUL is low.



(a) FD001

(b) FD004

(c) FD001

(d) FD004

Figure 6. Comparison of ESD and ENT as measures of uncertainty in terms of (a)-(b) Precision Recall Curves; and (c)-(d) F1 Scores with varying τ_u. ESD is a more robust uncertainty metric compared to ENT.

quantify uncertainty in the RUL estimates by using a simple average ensemble of the deep ordinal regression models. The proposed empirical standard deviation based metric for uncertainty provides accurate predictive uncertainty estimates: we observe low errors in RUL estimation for low uncertainty values. Further, the model is found to be accurate with high certainty when the remaining useful life is meager, i.e., the instance is approaching failure. It will be interesting to see if the ensemble based approach for uncertainty quantification can be extended to metric regression models as well using uncertainty methods for regression as proposed in (Lakshminarayanan et al., 2017).

The proposed approach for uncertainty estimation focuses on aleatoric uncertainty. To make the approach more practical, it is essential to include techniques that address epistemic uncertainty factors such as measurement noise and model process noise. Modelling epistemic uncertainty in deep learning models for timeseries has been recently studied in (Zhu & Laptev, 2017). They consider training an encoder-decoder framework to obtain latent representation of any given timeseries from the encoder. Once trained, the distance between any new given timeseries and the training set timeseries is calculated in the embedded space. If the calculated distance is found to be smaller, the prediction for the given new timeseries will be certain in nature and vice-versa. Moreover, for training an ensemble model for the purpose of uncertainty quantification, bagging can also be considered as an alternative approach where different random distributions of the training set are used to train different models. Further, the proposed metrics for uncertainty estimation need to be validated on other datasets such as battery failure as in (D. Liu, Luo, Peng, Peng, & Pecht, 2012) to establish their effectiveness and generalizability.

8. ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their insightful suggestions and feedbacks that help to improve the paper.

REFERENCES

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).

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2013). Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelligence*, *26*(7), 1751–1760.

Chang, K.-Y., Chen, C.-S., & Hung, Y.-P. (2011). Ordinal hyperplanes ranker with cost sensitivities for age estimation. In *Computer vision and pattern recognition (cvpr), 2011 ieee conference on* (pp. 585–592).

Cheng, J., Wang, Z., & Pollastri, G. (2008). A neural network approach to ordinal regression. In *Neural networks, 2008. ijcnn 2008.(ieee world congress on computational intelligence). ieee international joint conference on* (pp. 1279–1284).

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

Dong, H., Jin, X., Lou, Y., & Wang, C. (2014). Lithium-ion battery state of health monitoring and remaining useful life prediction based on support vector regression-particle filter. *Journal of power sources*, *271*, 114–123.

Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, *183*, 240–251.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Gugulothu, N., Malhotra, P., Vig, L., & Shroff, G. (2018). Sparse Neural Networks for Anomaly Detection in High-Dimensional Time Series. *AI4IOT Workshop at International Joint Conference on Artificial Intelligence (IJCAI)*.

Gugulothu, N., TV, V., Gupta, P., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2018). On practical aspects of using rnns for fault detection in sparsely-labeled multi-sensor time series..

Gugulothu, N., TV, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Predicting remaining useful life using time series embeddings based on recurrent neural networks. *International Journal on Prognostics and Health Management, IJPHM, arXiv preprint arXiv:1709.01073*.

Harrell, F. E. (2001). Ordinal logistic regression. In *Regression modeling strategies* (pp. 331–343). Springer.

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.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). Deepsurv: Personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, *18*(1), 24.

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265.

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.*

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems 30* (pp. 6402–6413).

Lam, J., Sankararaman, S., & Stewart, B. (2014). Enhanced trajectory based similarity prediction with uncertainty quantification. *PHM 2014*.

Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, *172*, 1–11.

Liu, D., Luo, Y., Peng, Y., Peng, X., & Pecht, M. (2012). Lithium-ion battery remaining useful life estimation based on nonlinear ar model combined with degradation feature. In *Annual conference of the prognostics and health management society* (Vol. 3, pp. 1803–1836).

Liu, H., Lu, J., Feng, J., & Zhou, J. (2017). Ordinal deep feature learning for facial age estimation. In *Automatic face & gesture recognition (fg 2017), 2017 12th ieee international conference on* (pp. 157–164).

Luck, M., Sylvain, T., Cardinal, H., Lodi, A., & Bengio, Y. (2017). Deep learning for patient-specific kidney graft survival analysis. *arXiv preprint arXiv:1705.10245*.

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 learning (icml 2016).*

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., 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).

Niu, Z., Zhou, M., Wang, L., Gao, X., & Hua, G. (2016). Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4920–4928).

Orozco, B. P., Abbati, G., & Roberts, S. (2018). Mordred: Memory-based ordinal regression deep neural networks for time series forecasting. *arXiv preprint arXiv:1803.09704*.

Park, L. A., & Simoff, S. (2015). Using entropy as a measure of acceptance for multi-label classification. In *International symposium on intelligent data analysis* (pp. 217–228).

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).

Sankararaman, S., Daigle, M., Saxena, A., & Goebel, K. (2013, March). Analytical algorithms to quantify the uncertainty in remaining useful life prediction. In *2013 ieee aerospace conference* (p. 1-11). doi: 10.1109/AERO.2013.6496971

Sankararaman, S., & Goebel, K. (2013). A novel computational methodology for uncertainty quantification in prognostics using the most probable point concept. In *Annual conference of the prognostics and health management society.*

Saurav, S., Malhotra, P., TV, V., Gugulothu, N., Vig, L., Agarwal, P., & Shroff, G. (2018). Online anomaly detection with concept drift adaptation using recurrent neural networks. In *Proceedings of the acm india joint international conference on data science and management of data* (pp. 78–87).

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. In *Prognostics and health management, 2008. phm 2008. international conference on* (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).

TV, V., Gupta, P., Malhotra, P., Vig, L., & Shroff, G. (2018). Recurrent neural networks for online remaining useful life estimation in ion mill etching system. In *Phm society conference* (Vol. 10).

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).

Yang, H.-F., Lin, B.-Y., Chang, K.-Y., & Chen, C.-S. (2013). Automatic age estimation from face images via deep ranking. *networks*, *35*(8), 1872–1886.

Yoon, A. S., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., ... Choi, Y. (2017). Semi-supervised learning with deep generative models for asset failure prediction. *arXiv preprint arXiv:1709.00845*.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, C., Lim, P., Qin, A. K., & Tan, K. C. (2016). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, *28*(10), 2306–2318.

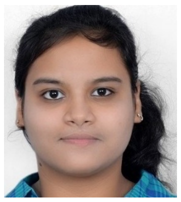
Zhang, Y., Xiong, R., He, H., & Pecht, M. (2018). Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *Prognostics and health management (icphm), 2017 ieee international conference on* (pp. 88–95).

Zhu, L., & Laptev, N. (2017). Deep and confident prediction for time series at uber. In *2017 ieee international conference on data mining workshops (icdmw)* (pp. 103–110).

BIOGRAPHIES

Vishnu TV is a Researcher in Deep Learning and Artificial Intelligence group at TCS Research, New Delhi, India for past 4 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.



Diksha is a Researcher in Deep Learning and Artificial Intelligence group at TCS Research, New Delhi, India for past 1 years. She has worked in Time series analytics and Recommender Systems using machine learning and deep learning based models. She completed her M.Tech in Computer Science from Indraprastha Institute of Information Technology, Delhi, India and B.Tech in Information Technology from Banasthali University, Rajasthan, India.

Pankaj Malhotra is a Researcher in Deep Learning and Artificial Intelligence group at TCS Research, New Delhi, India for past 8 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 and Artificial Intelligence 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, Time Series Analytics and Personalization.

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, he 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. APPENDIX

A.1. Performance Comparison between ORC and ORCE

As expected and summarized in Table 5, we observe that the ensemble of ORC models (i.e. ORCE, as used for the uncertainty quantification task), performs consistently better than ORC. Our MR implementation and the results are similar to LSTM-MR in (Zheng et al., 2017) (except, possibly for the choice of hyperparameters and train-validate splitting as the code to reproduce their results is not available). We observe that OR and ORC is not better than LSTM-MR for the full data scenario. However, in this work, we only intend to show the advantage of ORC for RUL estimation in highly censored scenarios with only a small number of failed instances.

A.2. Precision and Recall

Precision is the fraction of test instances with uncertainty below a threshold τ_u that also have error $\leq \tau_e$. Recall is defined as the fraction of test instances having uncertainty and error below some threshold τ_u and τ_e, respectively. More specifically:

$$\begin{aligned} Precision(P) &= \frac{\#(\hat{u} \leq \tau_u) \cap \#(|r - \hat{r}| \leq \tau_e)}{\#(\hat{u} \leq \tau_u)}, \\ Recall(R) &= \frac{\#(\hat{u} \leq \tau_u) \cap \#(|r - \hat{r}| \leq \tau_e)}{\#(test\ instances)}, \\ F1 &= 2 \times \frac{P \times R}{P + R} \end{aligned} \tag{11}$$

where $\#(X)$ denotes the number of instances satisfying the condition X.

A.3. Detailed Evaluation for Uncertainty Quantification

Instance Level Analysis: We perform a qualitative analysis of test instances having low uncertainty values despite having high error values to understand the scenarios where our proposed uncertainty quantification measure is failing. For such a test engine (test instance), we consider three nearest engines from the training set where nearness is defined in terms of Euclidean distance between the target vector estimate $\hat{\mathbf{y}}$ of test and training engines. After finding the nearest training engines, we plot the first PCA component corresponding to the test and respective nearest training engines.

For the PCA process, each multivariate reading from each timestamp is reduced to univariate reading by taking the first principal component, as discussed in (Malhotra, TV, et al., 2016).

PCA plot for test engine#93 from FD001 is shown in Figure 9(a). Although a large number of cycles have passed for this instance, it has a significantly high RUL. The entire life of this instance is 329, making it a rare example as instances with such high full cycles are not observed in the training data. Despite the passage of a higher number of cycles, RUL is very high. Training engines with such high RUL are rare, which is leading to higher error in spite of having low uncertainty.

Similarly, PCA plot for test engine#166 from FD004 is shown in Figure 9(b). Due to the assumption of having maximum RUL r as 130 in ORC formulation, predicted life is around 130, even though the actual life is significantly higher. This clipping effect causes high error. Moreover, scarcity of training engines with such high RUL further leads to an increase in error in spite of having low uncertainty.

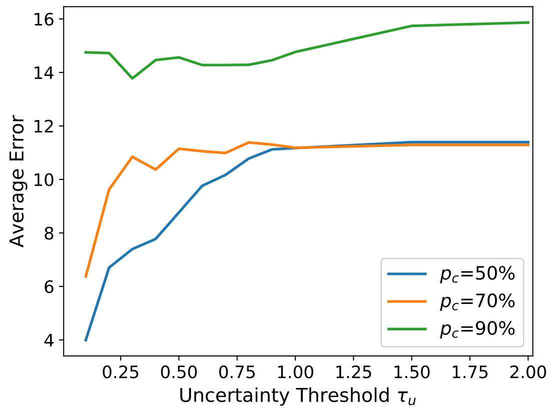
Uncertainty Evaluation wrt Error: We expect our model to be highly uncertain for higher error values. To evaluate the same, we calculate C_e as follows:

$$C_e = \frac{\#(|r - \hat{r}| \leq \tau_e) \cap \#(\hat{u} \leq \tau_u)}{\#(|r - \hat{r}| \leq \tau_e)} \tag{12}$$

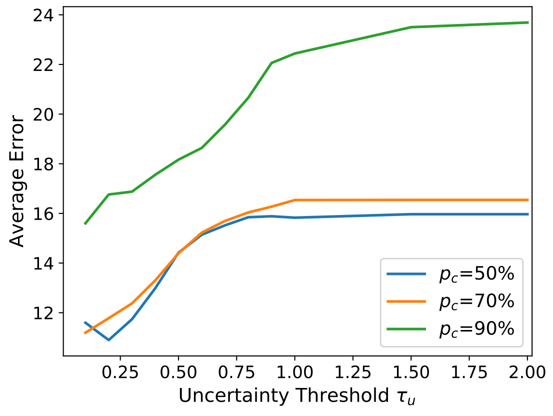
where, C_e is the fraction of test instances with error and uncertainty $\leq \tau_e$ and $\leq \tau_u$ respectively.

We compute C_e at varying error threshold, $10 \leq \tau_e \leq 130$ and fixing uncertainty threshold τ_u as 0.2. The results are shown in Figure 11. At lower τ_e, higher C_e indicates that a higher fraction of correct predictions is confident in nature.

Relationship Between Error and Uncertainty: RUL estimates with lower error values are expected to be certain in nature. For evaluating the uncertainty quantification metric from this aspect, we plot the relationship between error and average uncertainty, shown in Figure 10. Average uncertainty value at given τ_e is computed by considering the test instances with RUL estimate error $\leq \tau_e$ and we then average out the uncertainty values corresponding to these filtered out test instances. We observe that at lower error thresholds, the computed average uncertainty is also low, indicating the preciseness of the RUL estimation model. Further, the increase in average uncertainty with an increase in error threshold indicates reliable behavior of the RUL estimation model.

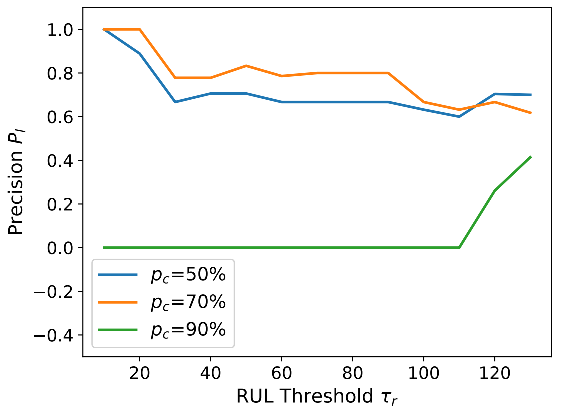


(a) FD001 Dataset

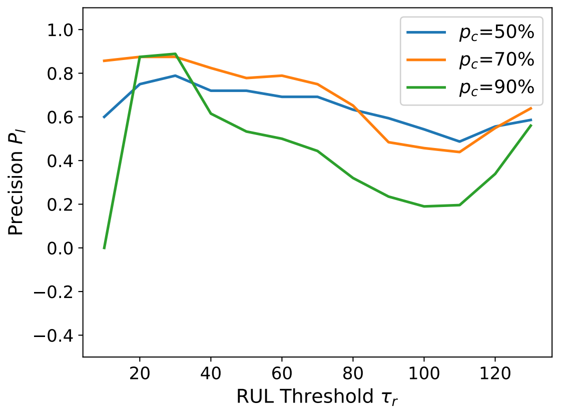


(b) FD004 Dataset

Figure 7. Average Error at varying τ_u.

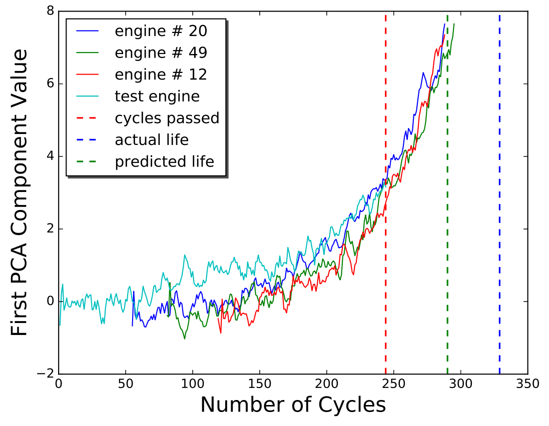


(a) FD001 Dataset

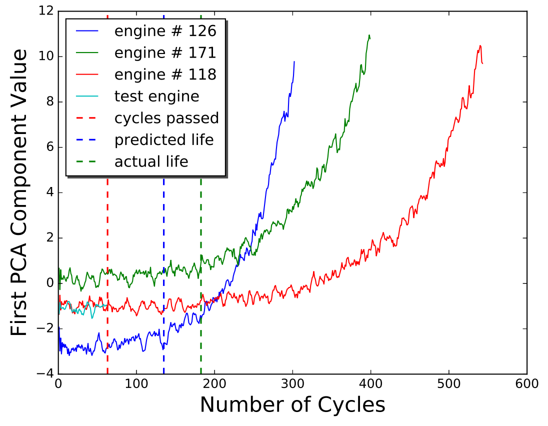


(b) FD004 Dataset

Figure 8. Uncertainty Evaluation wrt RUL.

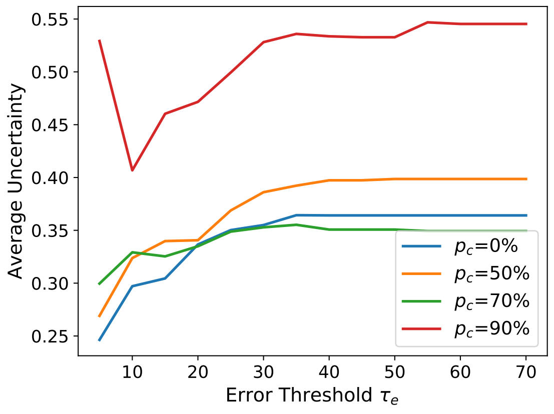


(a) FD001 Dataset

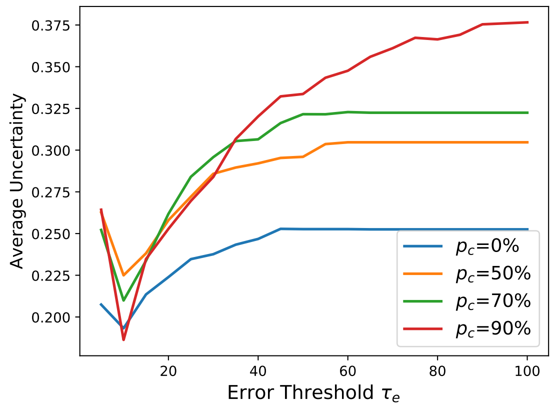


(b) FD004 Dataset

Figure 9. Instance Level Analysis (PCA) showing instances with low uncertainty estimate but high error in RUL estimate.



(a) FD001 Dataset

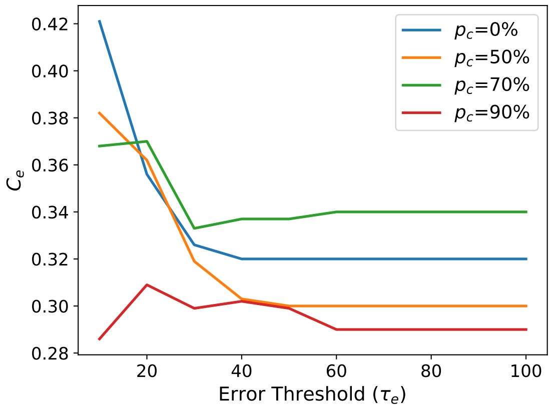


(b) FD004 Dataset

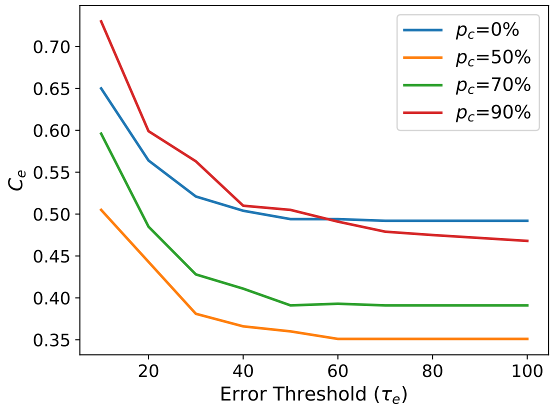
Figure 10. Average Uncertainty with varying τ_e indicating low uncertainty values when error in RUL estimates is low.

	FD001						FD004					
	Instances		RMSE		Timeliness Score (S)		Instances		RMSE		Timeliness Score (S) $\times 10^{-3}$	
$p_c(\%)$	n_f	n_c	ORC	ORCE	ORC	ORCE	n_f	n_c	ORC	ORCE	ORC	ORCE
0	80	0	15.63	**14.62**	367.64	**292.76**	199	0	28.33	**27.47**	6.44	**5.24**
50	40	40	17.60	**15.98**	572.63	**372.26**	100	99	31.48	**30.62**	9.83	**7.86**
70	24	56	18.53	**16.57**	561.11	**404.94**	60	139	32.13	**31.27**	12.0	**9.59**
90	8	72	21.51	**20.38**	20.64×10^2	$\mathbf{13.57 \times 10^2}$	20	179	39.75	**38.41**	141.13	**60.72**

Table 5. Comparison of ORC and ORCE considered in terms of RMSE and Timeliness Score (S) for FD001 and FD004 datasets. n_f and n_c denote number of failed and censored instances in training set, respectively.



(a) FD001 Dataset



(b) FD004 Dataset

Figure 11. Uncertainty Evaluation w.r.t. Error.