Reliability Engineering and System Safety 196 (2020) 106727

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Controlling the accuracy and uncertainty trade-off in RUL prediction with a surrogate Wiener propagation model

Yingjun Deng[a,b,1,*], Alessandro Di Bucchianico[b], Mykola Pechenizkiy[b]

[a] Center for Applied Mathematics, Tianjin University, No.92 Weijin Road, Tianjin 300072, China
[b] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven 5600 MB, Netherlands

ARTICLE INFO

Keywords:
Remaining useful lifetime
Uncertainty propagation
Recurrent neural network
Long short-term memory
Wiener process
Surrogate modeling

ABSTRACT

In modern industrial systems, sensor data reflecting the system health state are commonly used for the remaining useful lifetime (RUL) prediction, which are increasingly processed by modern deep learning based approaches recently. But these deep learning models do not automatically provide uncertainty information for the RUL prediction, hence this paper is motivated to introduce a novel approach that allows to control trade-off between prediction performance and knowledge about the uncertainty of the RUL prediction. The key aspect of our approach is to use a long short-term memory (LSTM) network as an expressive black-box predictor and the Wiener process as a surrogate to model the propagation of prediction uncertainty. The uncertainty propagation model is used to interactively train the RUL predictor. Our empirical results in a turbofan engine degradation simulation use case show that the surrogate Wiener propagation model can improve the near-failure prediction accuracy by sacrificing the far-to-failure prediction accuracy.

1. Introduction

Remaining useful lifetime (RUL) is a key performance indicator in modern industrial systems, leading to automated decisions for maintenance, production, and inventory [1–3]. Due to advances in sensor engineering and monitoring techniques, more and more inspection data have become available. These inspection data indirectly reflect the system health state, which are commonly used for RUL prediction.

However in real-life applications, establishing data-driven RUL predictors for industrial systems faces several challenges. Common, well-known challenges are the following.

- High-dimensional, noisy inspection data come from various sensors at different positions, including vibration, acceleration, temperature [4], acoustic signals [5] etc.
- Unknown failure mechanisms prevent the degradation indicator extraction directly.
- As run-to-failure tests need a long running period and also a high economic cost, usually few historical failure records exists.

In this paper we focus on the challenge of real-time prediction in the common situation in which high-dimensional inspection data come sequentially and all inspection data are associated to the same system failure. In such cases recorded RUL values linearly decrease over time, and hence the RUL prediction trained with RUL records fits better a linear propagation. However, black-box prediction models based on historical data do not automatically yield linear trends and thus may have poor performance. How to keep the rationale about prediction propagation while facing the curse of dimensionality motivates our research in this paper.

Traditional RUL prediction models, like stochastic processes [1,6], mixed-effect physical models [7], and proportional hazard rate models [8,9] based on manual feature engineering success in many scenarios with scalar degradation data[1], but they usually fail to perform well in high-dimensional, high velocity inspection data settings. This demand largely drives recent advances in machine learning, especially in deep neural networks.

Earlier trials based on artificial neural networks (ANNs) [10–15] with shallow structures did not yield satisfactory results, but recently developed deep structures seem to be very promising. For instance, recurrent neural networks (RNN) and related variants [16–18] are introduced as a standard way to process time-stamped sensor readings, and convolutional neural networks (CNN) [19,20] showing excellent capability of representing degradation have also become popular. In particular, deep neural networks show a great success for self-learning feature representation [21] and are becoming increasingly popular in

* Corresponding author.
E-mail addresses: yingjun.deng@tju.edu.cn (Y. Deng), A.d.Bucchianico@tue.nl (A.D. Bucchianico), m.pechenizkiy@tue.nl (M. Pechenizkiy).
[1] Current address: Center for Applied Mathematics, Tianjin University, No.92 Weijin Road, Tianjin 300072, China.

https://doi.org/10.1016/j.ress.2019.106727
Received 31 January 2019; Received in revised form 21 September 2019; Accepted 31 October 2019
Available online 04 November 2019

reliability engineering.

A main drawback to using machine learning models for RUL prediction is the lack of interpretability and transparency. This difference with explainable statistical and stochastic methods in traditional reliability engineering prevents the machine learning models to be widely applied when uncertainty quantification and decision analysis are needed. More importantly, reliable uncertainty quantification requires parametric probabilistic models and it is not obvious how to connect these models with RUL predictions based on machine learning methods.

To address the above concern, in this paper we refer to the idea of surrogate modeling to find a simplified explainable model that approximates multivariate complex systems based on limited historical data [22]. The main contribution of this paper is to introduce a latent Wiener process as the bridge process between machine learning based RUL predictions and parametric probabilistic models for uncertainty quantification. The modeling adopts a backward strategy to model the predictor's output rather than the forward strategy to model the predictor's input. Our rationale is to first assume a Wiener propagation model for the optimal output, and then optimize the predictor to fit the surrogate Wiener model.

The use of an unobserved bridge process approach is widely adopted in surrogate modeling [23,24], or latent process modeling [25,26] for hidden states. Specifically in the literature regarding RUL prediction, the surrogate modeling or latent process modeling is commonly performed to find the scalar degradation or health indicator of systems, which is called stochastic degradation modeling [1,27]. In high-dimensional inspection scenarios, the scalar degradation indicator usually cannot be directly observed, leading to the problem of missing output labels during surrogate modeling. Instead we do surrogate modeling for RUL prediction in this paper, that can be evaluated directly by failure records. Moreover the idea of combining machine learning with stochastic models has been applied to other problems in lifetime analysis. E.g, [28] use machine learning is used to learn the structure of Bayesian networks and apply the Bayesian networks for fault diagnosis in engineering while in [29] proportional hazards models are combined with neural networks.

This paper is organized as follows. A Wiener propagation model for prediction increments will be introduced in Section 2, where the joint density to observe the prediction and the observation is expressed explicitly. In Section 3, the RUL predictor and the Wiener propagation model will be jointly trained based on negative likelihood loss functions. A long short-term memory network (LSTM) predictor will be specified in Section 4 and it will be verified in a turbofan engine degradation simulation use case. Conclusions will be given at the end.

2. Wiener propagation model

2.1. Problem setup

All inspection times are assumed to be non-negative integers, unless otherwise specified. Consider an unspecified stochastic system equipped with m sensors. Each inspection at an integer time $s \in \mathbb{N}$ returns a vector $\mathbf{x}_s \in \mathbb{R}^m$. The system fails at time $\tau \geq 0$. All inspection data observed in an interval $[s, t]$ are denoted by $\mathbf{x}_{s:t}$, and the RUL at time t is denoted by $r_t = \tau - t$ for $t < s < \tau$. We assume that an RUL predictor φ is available which maps $\mathbf{x}_{s:t}$ to an estimate of r_t, say $\alpha^{[s:\,t]}$,

$$\begin{aligned} &\varphi: \mathbb{R}^{(t-s+1)\times m} \to \mathbb{R}, \\ &\text{with } \varphi(\mathbf{x}_{s:t}) = \alpha^{[s:t]},\ 0 \leq s < t < \tau. \end{aligned} \tag{1}$$

We will write α_s instead of $\alpha^{[0:\,s]}$ for short. Note that for $s \geq \tau$ there is no prediction needed, because the failure has already occurred.

Since we deal with sequential RUL prediction, the predictor φ is assumed to have following properties throughout this paper.

1. The RUL predictors are stationary.
2. Differences between consecutive RUL predictions are normally distributed with average decrease of one time unit, i.e.
$$\varphi(\mathbf{x}_{0:s+1}) - \varphi(\mathbf{x}_{0:s}) \sim \mathcal{N}(-1, \sigma^2) \tag{2}$$
for some $\sigma > 0$, where the notation $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2, whose probability density function is denoted by $\mathcal{N}(x; \mu, \sigma^2)$ for $x \in \mathbb{R}$.
3. The predictor can iteratively update the old prediction by new information.

These rough assumptions will be specified and deepened in a more rigorous setting in later discussions. The first and second property lead to the Wiener propagation modeling in this section. The third iterative property fits well the LSTM predictor [30] that will be discussed in Section 4. Without loss of generality, the available information in practice is set up as follows.

- Historical inspection data on N identical and independent systems are observed with associated failure times $\{\tau_i\}_{i=1}^N$.
- Suppose there are totally n_i inspections for the ith system until the failure time τ_i. The inspection data at time s_{ij}, $j = 1, \ldots, n_i$ are denoted by $\mathbf{x}_{\mathbf{s_{ij}}}$; the associated RUL at time s_{ij} for the ith system is observed by $r_{sij} = \tau_i - s_{ij}$.
- Due to unsynchronized calendar times in different systems, initial states in different systems may be different.

As the stochastic system is considered, the prediction α_s depending on uncertain system states in (s, τ) cannot be almost surely accurate until the failure happens. Hence the prediction uncertainty exists and cannot be compressed. However a good prediction can redistribute uncertainties at different times and focus on the average prediction capacity. In the next section, what a good prediction looks like will be discussed.

2.2. Uncertainty propagation modeling

Due to unobserved system uncertainties[31], the system state $\mathbf{x}_s$ are uncertain on different systems leading to the RUL prediction uncertainty. Suppose the RUL prediction α_s, $s \in [0, \tau)$ on an unspecified system is given, this section contributes to the uncertainty expression about $\alpha_t - r_t$, at a previous time $s < t$.

In the real-time prediction at time $s < t$, the prediction α_t and the actual RUL r_t are not actually observed. Hence the modeling is divided into two parts based on the existing prediction α_s: prediction propagation modeling for $\alpha_t - \alpha_s$, $t > s$ and prediction error modeling $\alpha_s - r_s$ when r_s is given. In the following, the propagation modeling will be done via a Wiener process, and the validation error $\alpha_s - r_s$ will be processed as normalized random variables.

2.2.1. An illustrative RUL predictor under the scalar Wiener degradation process

Before proceeding to later discussions, let us recall some preliminary results on Wiener processes in a classical degradation modeling view [1,27,32]. Firstly, the inverse Gaussian distribution [33] is introduced as follows.

Definition 2.1. The inverse Gaussian distribution has a probability density function $\mathcal{G}(\cdot;\alpha, c)$ with parameters $\alpha > 0$ and $c > 0$ given by

$$\mathcal{G}(x; \alpha, c) = \left[\frac{c\alpha^2}{2\pi x^3}\right]^{1/2} \exp\left\{\frac{-c(x-\alpha)^2}{2x}\right\},\ x > 0. \tag{3}$$

If a random variable X follows the inverse Gaussian distribution denoted by $X \sim \mathcal{G}(\alpha, c)$, then

$$\mathbb{E}(X) = \alpha,\ \mathrm{Var}(X) = \alpha/c. \tag{4}$$

It is noted that notations in Definition 2.1 are different from other definitions (e.g. [33]) to facilitate later discussions.

Now consider a scalar drifted Wiener process [34] for the scalar inspection data scenario in (1). The inspection data $\mathbf{x_t}$ at time $t \geq 0$ are observed from

$$X_t = -\mu t + \sigma W_t, \tag{5}$$

where $X_0 = \mathbf{x_0} > 0$, μ, $\sigma > 0$ and W_t is a standard Wiener process [34].

As a standard way to model failures, the first passage time of X_t to reach the boundary 0 is adopted as failure time,

$$T_{\mathbf{x_0}} := \inf_{t\geq 0}\{X_t \leq 0 | X_0 = \mathbf{x_0}\}. \tag{6}$$

The probability distributions of $T_{\mathbf{x_0}}$ is explicitly given by the following proposition.

Proposition 2.2. *With notations defined in (5) and thereafter, given $X_t = -\mu t + \sigma W_t$, $T_x = \inf_{t\geq 0}\{X_t \leq 0 | X_0 = x\}$ follows the inverse Gaussian distribution defined in (3)*

$$T_x \sim \mathcal{G}(x/\mu, \mu^2/\sigma^2). \tag{7}$$

Proof. It is a classical result that the first passage time of a drifted Wiener process to a constant boundary follows the inverse Gaussian distribution. Hence the proof is omitted, and readers can refer to other literature, e.g. Section 3.1.4, [34]. □

Now we turn to RUL prediction from observation to the degradation process X_t, $t \geq 0$. Suppose until time s, the system is not failed with the observation $X_{0:s} = \mathbf{x_{0:s}}$. The RUL prediction is naturally defined as the conditional expectation $\hat{R}_s := \mathbb{E}(T_{\mathbf{x_0}} - s | X_s = \mathbf{x_s}, T_{\mathbf{x_0}} \geq s)$. From the stationary and Markov property of Wiener process, $\hat{R}_s := \mathbb{E}(T_{\mathbf{x_0}} - s | X_s = \mathbf{x_s}, T_{\mathbf{x_0}} \geq s) \simeq T_{\mathbf{x_s}}$ holds.

Moreover from Proposition 2.2, $\hat{R}_s \simeq T_{\mathbf{x_s}} \sim \mathcal{G}(\mathbf{x_s}/\mu, \mu^2/\sigma^2)$. The value-oriented RUL predictor φ in (1) can be naturally specified as the expectation of $\hat{R}_s$ under the Wiener degradation setting, and the associated prediction α_s equals

$$\alpha_s := \varphi(\mathbf{x_{0:s}}) = \mathbb{E}(\hat{R}_s) = \mathbb{E}(T_{\mathbf{x_s}}) = \frac{\mathbf{x_s}}{\mu}. \tag{8}$$

Hence with observation to (5), an illustrative predictor $\varphi(\mathbf{x_{0:s}}) = \mathbf{x_s}/\mu$ as a linear function of the newest observation is presented. This also summarizes basic procedures of RUL prediction based on verified stochastic degradation process [1].

However the above scalar scenario does not work for general multi-dimensional inspection data in (1) where the health indicator is hidden behind the inspection data. This leads to the idea to introduce a bridge process between RUL prediction and inspection data in the next subsection.

2.2.2. Surrogate Wiener propagation model

If a validation data-set is given, the prediction error in the validation test shows the intrinsic uncertainty caused by the predictor. Moreover the prediction is more accurate, it is more possible to observe r_s and α_s simultaneously. Hence to model the prediction uncertainty in a probabilistic view, the possibility to observe the prediction and the observation (r_s, α_s) simultaneously motivates later discussions. This possibility relationship can hardly be revealed directly due to the unknown failure mechanism.

In this paper a hidden bridge stochastic process Y_s observed by the prediction α_s in (1) will be introduced to model the joint likelihood of (r_s, α_s). From the backward-validation view, $Y_s = \alpha_s$ as a prediction can be naturally evaluated by r_s. From the forward-prediction view, the RUL prediction $\hat{R}_s$ conditional on $Y_s = \alpha_s$ can be derived explicitly by introducing Y_s, which is observed by the actual RUL r_s. Hence,the possibility to observe (r_s, α_s) simultaneously can be expressed as the joint density to observe $\hat{R}_s = r_s$ and $Y_s = \alpha_s$,

$$possibility(r_s, \alpha_s) = p_{\hat{R}_s}(r_s | Y_s = \alpha_s) p_{Y_s}(\alpha_s). \tag{9}$$

The basic bridge idea is adopted widely in the literature under different settings. For instance, Bengio et al. used the hidden Markov model as the bridge process between acoustic signals and speech texts in [26], and Christophe et al. used Gamma process as the bridge process between the degradation signal and the RUL prediction in [35].

The remaining of this section will reveal the hidden process Y_s explicitly under moderate assumptions to specify the considered scenario.

Assumption 2.3.

1. The system degrades gradually.
2. The system failure is detected immediately and perfectly.
3. Newer and more information contributes to less absolute prediction error.

Assumption 2.3 is ansatz to formulate later modeling, but they are also proposed from observed facts and general rationale in practical scenarios where at least Wiener degradation processes are applicable [32,36]. Especially,

- Point 1) in Assumption 2.3 means that the system degradation states have no large jumps over time, for instance early-stage fatigue cracks usually fits this assumption in practice (see illustrative examples in [37]).
- Point 2) in Assumption 2.3 means that failure definition and detection are perfect, which fits the case of hard failures leading to the immediate system shutdown.
- Point 3) in Assumption 2.3 is the rationale of data-driven methodology to do RUL prediction.

In the following we will introduce Wiener process to model the RUL prediction propagation, which is motivated by our real-world experience on general prediction error analysis. Irrelevant to specific prediction techniques, the RUL prediction usually show following properties.

- Prediction errors are time-dependent and mutually dependent.
- Prediction errors at a time are Gaussian, or normally distributed.
- RUL prediction as a time-to-event prediction has a linear trend over time.

The above rationale combining with Assumption 2.3 supports us to model the propagation for RUL prediction via Wiener process [34] based on its natural relationship with white noises. Specifically recalling the RUL predictor (1), the contribution of $\mathbf{x_{s:t}}$ to the prediction increment $\alpha_t - \alpha_s$ is modeled as the increment of a Wiener process in the following assumption.

Assumption 2.4.

1. For two times $t \geq s$, the RUL prediction α_t uses more data $\mathbf{x_{s:t}}$ than α_s, the information difference contributes to uncertainty in the prediction increment $\alpha_t - \alpha_s$. Hence the uncertainty of prediction increment is assumed to be linearly related with $t - s$.
2. The prediction increment is memoryless, unbiased and cumulating over the time difference. Combining the last assumption, for $t \geq s \geq 0$ and $c > 0$, the prediction increment is assumed to be observed from the increment of a Wiener process, i.e.

$$\alpha_t - \alpha_s \simeq s - t + \frac{W_{t-s}}{\sqrt{c}}, \tag{10}$$

   where W_s is a standard Wiener process [34].
3. As Wiener process follows the normal distribution, the RUL prediction increment $\alpha_t - \alpha_s$ for $t \geq s$ is independent with α_s and normally distributed with mean $s - t$ and variance $(t - s)/c$,

$$\alpha_t - \alpha_s \sim \mathcal{N}(s-t, (t-s)/c). \tag{11}$$

If there is no uncertainty in the predictor φ, the increment $\alpha_t - \alpha_s$ as the RUL difference should be exactly $s - t$. Hence (10) reasonably models the cumulating uncertainty in $\alpha_t - \alpha_s$ and the parameter c controls the uncertainty propagation rate. It is also noted the increment modeling does not depend on the calendar time, but only the time difference. This leads to two different views on the forward prediction in the calendar time and the backward validation over the actual RUL respectively.

2.2.3. Backward validation

Directly from the propagation model (10) a natural backward validation model is implied by Assumption 2.4, leading to the following proposition. This is designed for the post-failure analysis when failures are already observed.

Proposition 2.5. *If the failure time has been observed at time $\tau \geq 0$ with the actual RUL $r_s = \tau - s$ for $s \in [0, \tau]$, the historical prediction α_s under Assumption 2.4 is observation for a Wiener process $r_s + W_{r_s}/\sqrt{c}$ where W_s is a standard Wiener process, and $c > 0$ defined in (10).*

Moreover, α_s follows normal distribution with mean r_s and variance r_s/c, i.e.

$$\alpha_s \sim \mathcal{N}(r_s, r_s/c). \tag{12}$$

Proof. From Assumption 2.3, the failure is detected immediately, so at the failure time τ the prediction should be accurate such that $\alpha_\tau = r_\tau = 0$. Furthermore for two times $\tau > s \geq 0$, $r_\tau - r_s = s - \tau$ (11) leads to

$$\alpha_\tau - \alpha_s \sim \mathcal{N}(r_\tau - r_s, (r_s - r_\tau)/c). \tag{13}$$

And from the time reversal property of standard Wiener process [34], (10) leads to

$$\alpha_\tau - \alpha_s \simeq r_\tau - r_s + \frac{W_{r_s - r_\tau}}{\sqrt{c}}. \tag{14}$$

Let $\alpha_\tau = r_\tau = 0$, (12) and the corresponding process comes naturally. □

It is remarked that Propagation 2.5 is consistent with Assumption 2.3 as more accuracy is present when the time is closer to the failure time.

2.2.4. Forward prediction

Now consider the forward RUL prediction when the system failure is not observed yet, and the target to estimate the prediction uncertainty when the RUL prediction α_s is given at time s. From Eq. (10), for an unspecified system the RUL prediction α_s is observed from the drifted Wiener process Y_s,

$$Y_s = -s + \frac{1}{\sqrt{c}} W_s,\ s \geq 0 \tag{15}$$

with the initial value $Y_0 = \alpha_0$. The above model predicts the value of α_s, $s \geq 0$ based on the initial prediction α_0. More importantly, Y_s globally describes the time-dependent uncertainty and the dependence of the predictor φ in a probabilistic way.

- Now consider the failure prediction based on Eq. (15) at time 0 with $Y_0 = \alpha_0$. Noticing Assumption 2.3 implies $\alpha_\tau = 0$, the failure time can be naturally processed as the realization of the first passage time for Y_s to reach 0 based on (15),

$$T_{\alpha_0} = \inf_{t \geq 0}\{Y_t \leq 0 | Y_0 = \alpha_0\},\ \ \alpha_0 \geq 0. \tag{16}$$

  If $\alpha_0 < 0$, the predicted failure time equals 0 without uncertainty.
- When $Y_s = \alpha_s \geq 0$ is observed at time $s > 0$, it implies that the system is not failed in the failure model(16), i.e. $T_{\alpha_0} \geq s$. The RUL prediction is updated by the conditional expectation under the process (15),

$$\hat{R}_s = \mathbb{E}(T_{\alpha_0} - s | Y_s = \alpha_s,\ T_{\alpha_0} \geq s) \simeq T_{\alpha_s},\ \alpha_s \geq 0. \tag{17}$$

- From Proposition 2.2,

$$T_{\alpha_0} \sim \mathcal{G}(\alpha_0, c),\quad \textit{and}\quad \hat{R}_s \sim \mathcal{G}(\alpha_s, c). \tag{18}$$

The above hidden failure modeling is self-consistent based on Y_s. And the prediction uncertainty of α_s to fit the actual RUL is expressed in (18).

2.2.5. Joint-likelihood of observation and prediction

Recalling the target (9), the possibility to observe r_s and α_s simultaneously is now explicitly modeled by introducing the propagation model Y_s (15) and Proposition 2.5. This connection is described explicitly via density functions of Y_s and $\hat{R}_s$.

- For $\alpha_s = \varphi(\mathbf{x}_{0:s})$, $Y_s = \alpha_s$ is the initial value for Y_t, $t \geq s$ satisfying the propagation model (15). And the density function of Y_s is revealed by the normal distribution based on Proposition 2.5,

$$p_{Y_s}(\alpha_s) = \mathcal{N}(\alpha_s; r_s, \frac{r_s}{c}). \tag{19}$$

  For two inspection times $t > s$ regarding the same system with predictions α_t and α_s, the independent increment and stationarity of Wiener process leads to $Y_t - Y_s \sim \mathcal{N}(s-t, (t-s)/c)$ with the transition density

$$p_{Y_t}(\alpha_t | Y_s = \alpha_s) = \mathcal{N}(\alpha_t; \alpha_s + s - t, (t-s)/c). \tag{20}$$

- At time s with the prediction α_s, the conditional expectation of $\hat{R}_s$ in (18) describes the RUL prediction based on Y_t, $t \geq s$. Hence the density function to observe the actual RUL r_s is

$$p_{\hat{R}_s}(r_s | Y_s = \alpha_s) = \mathcal{G}(r_s; \alpha_s, c). \tag{21}$$

From (18), (19), (21), the following expression summarizes the connection between r_s and α_s,

$$\begin{aligned} &possibility(r_s, \alpha_s) = p_{\hat{R}_s, Y_s}(r_s, \alpha_s) \\ &= p_{\hat{R}_s}(\alpha_s | Y_s = \alpha_s) p_{Y_s}(\alpha_s) \\ &= \mathcal{G}(r_s; \alpha_s, c)\mathcal{N}(\alpha_s; r_s, r_s/c). \end{aligned} \tag{22}$$

(22) clearly shows the dependence between r_s and α_s via the hidden process (15).

It is noted that RUL predictions based on sequential inspection data $\mathbf{x}_{0:s}$ are mutually dependent at different times in the same system due to the common knowledge on historical inspection data. Hence in this scenario for two times $t > s$, the possibility to observe $(r_t, \alpha_t, \alpha_s)$ simultaneously at time t models the joint prediction uncertainty. From the Markov property of Y_s,

$$\begin{aligned} &possibility(r_t, \alpha_t, \alpha_s) \\ &= p_{\hat{R}_t}(r_t | Y_t = \alpha_t) p_{Y_t}(\alpha_t | Y_s = \alpha_s) p_{Y_s}(\alpha_s) \\ &= \mathcal{G}(r_t; \alpha_t, c)\mathcal{N}(\alpha_t - \alpha_s; s - t, \frac{t-s}{c})\mathcal{N}(\alpha_s; r_s, \frac{r_s}{c}). \end{aligned} \tag{23}$$

This treatment can be similarly applied to the scenario that multiple dependent predictions exist.

The above probabilistic understanding between α_s and r_s leads to the predictor estimation via maximizing the likelihood in the next section.

3. Predictor training

Given the RUL predictor $\varphi(\cdot)$, a new parameter c is introduced to control the prediction uncertainty in the last section. In this section, how to estimate this new parameter on the given training data and its role to improve the prediction loss will be investigated.

3.1. Training objectives

In the last section, a Wiener propagation model is introduced. The RUL prediction based on α_s is modeled via a latent Wiener degradation process (15) and the RUL model (18). This section contributes to jointly optimize the RUL prediction (18) and the hidden process (15).

For an unspecified system at time s, the actual observation $(\mathbf{x}_{\mathbf{0:\,s}}, r_s)$ forms an inspection-RUL pair. Moreover a RUL prediction α_s bridges the inspection and the RUL prediction, hence a triple $(\mathbf{x}_{\mathbf{0:\,s}}, \alpha_s, r_s)$ is constructed. Under the setting of (22), the following discussions aim to maximize the possibility to observe this triple.

From (1) and (15), two pending parameters Θ, c regarding the RUL prediction are to be estimated under the following setting.

1. Suppose the prediction α_s comes from a predictor φ with the pending vector parameter Θ, i.e. $\alpha_s = \varphi(\mathbf{x}_{\mathbf{0:s}}; \Theta)$.
2. α_s is the observation of Y_s in (15) with the pending parameter c.
3. From (18), the actual RUL r_s is observation of $\hat{R}_s$ following the inverse Gaussian distribution, i.e. $\hat{R}_s \sim \mathcal{G}(\alpha_s, c)$.

From (22), the optimization on Θ, c aims to better describe actual RUL records, and to better fit the hidden process (15). These two objectives are different and dependent, which are connected by the hidden process Y_s in (15). Hence the prediction can be evaluated by the joint log-likelihood to observe $\alpha_s = \varphi(\mathbf{x}_{\mathbf{0:s}}; \Theta)$ and r_s from (22). The prediction loss is naturally defined as the negative log-likelihood.

- The prediction loss at time s with prediction α_s for the actual RUL r_s without historical inspections is defined by

$$\begin{aligned} &error(r_s, \alpha_s; \Theta, c) \\ &= -2[\log p_{\hat{R}_s}(r_s|Y_s = \alpha_s) + \log p_{Y_s}(\alpha_s)] \\ &= -2[\log \mathcal{G}(r_s; \alpha_s, c) + \log \mathcal{N}(\alpha_s, r_s, r_s/c)]. \end{aligned} \tag{24}$$

  Here the constant 2 is introduced for computation convenience without influence to optimization.
- On the same system, if different predictions α_{s_j} are given at s_j, $j = 1, \ldots$, the loss function can be set up based on (23) to tolerate the dependence,

$$\begin{aligned} &error(r_{s_{j+1}}, \alpha_{s_{j+1}}, \alpha_{s_j}, \ldots \alpha_{s_1}; \Theta, c) = \\ &-2\left[\log p_{R_{s_{j+1}}^{\wedge}}(r_{s_{j+1}}|Y_{s_{j+1}} = \alpha_{s_{j+1}}) + \log p_{Y_{s_1}}(\alpha_{s_1}) \right. \\ &\left. + \sum_{i=2}^{j+1} \log p_{Y_{s_l}}\left(\alpha_i|Y_{s_{l-1}=\alpha_{s_{l-1}}}\right)\right], j = 1, 2\ldots \end{aligned} \tag{25}$$

For all available data specified in Section 1, the total loss is defined as the average loss for all observed inspection-RUL pairs $\{(\mathbf{x}_{\mathbf{0:s_{ij}}}, r_{s_{ij}})\}_{j=1}^{n_i}$, $i = 1, \ldots, N$ which are denoted simply as D_{train} for training in later discussions.

These data pairs contribute to the following prediction loss function from (24) and (25) with the notation $\Delta_{ij} = s_{i(j+1)} - s_{ij}$, $\alpha_{ij} = \varphi(\mathbf{x}_{\mathbf{0:s_{ij}}}; \Theta)$,

$$\begin{aligned} &L_p(\Theta, c; D_{train}) \\ &= \frac{1}{N}\sum_{i=1}^{N} error\left(r_{s_{in_i}}, \alpha_{in_i}, \ldots, \alpha_{i1}; \Theta, c\right) \\ &= \frac{-2}{N}\sum_{i=1}^{N}\left[\log \mathcal{G}\left(r_{s_{in_i}}; \varphi\left(\mathbf{x}_{\mathbf{0:s_{in_i}}}; \Theta\right), c\right)\right. \\ &+ \log \mathcal{N}(\varphi(\mathbf{x}_{\mathbf{s_{i1}}}; \Theta); r_{s_{i1}}, \frac{r_{s_{i1}}}{c}) \\ &\left. + \sum_{j=1}^{n_i-1} \log \mathcal{N}(\varphi(\mathbf{x}_{\mathbf{s_{i(j+1)}}}; \Theta) - \varphi(\mathbf{x}_{\mathbf{s_{ij}}}; \Theta); -\Delta_{ij}, \frac{\Delta_{ij}}{c})\right] \end{aligned} \tag{26}$$

Hence the optimal parameters Θ^*, c^* can be solved directly from the minimization problem on the training data D_{train},

$$(\Theta^*, c^*) = \arg\min_{\Theta, c} L_p(\Theta, c; D_{train}). \tag{27}$$

When the pending vector parameter Θ consists of a few elements, the above optimization problem can be solved generally by self-designed optimization algorithms. However for a RUL predictor with thousands and more pending parameters, e.g. a deep neural network, to rewrite the optimization algorithm on the predictor such that the new parameter c can be tolerated is a waste of existing training experience. Hence in the following, the optimization will be discussed based on Θ and c respectively.

3.2. Loss redefinition

With explicit density functions defined in (3), directly differentiating the loss function (26) with respect to c leads to Eq. (28).

$$\begin{aligned} &\frac{\partial L_p(\Theta, c; D_{train})}{\partial c} = \\ &\frac{1}{N}\sum_{i=1}^{N}\left[-\frac{1}{c} + \frac{(r_{s_{in_i}} - \alpha_{in_i})^2}{r_{s_{in_i}}} - \frac{1}{c} + \frac{(\alpha_{i1} - r_{s_{i1}})^2}{r_{s_{i1}}}\right. \\ &\left. + \sum_{j=1}^{n_i-1}(-\frac{1}{c} + \frac{(\alpha_{i(j+1)} - \alpha_{ij} + \Delta_{ij})^2}{\Delta_{ij}})\right]. \end{aligned} \tag{28}$$

Let $\frac{\partial L_p(\Theta, c; D_{train})}{\partial c} = 0$, naturally the unique optimal value of c regarding the pending Θ is returned with notations α_{ij} and Δ_{ij} in (26),

$$\gamma(\Theta; D_{train}) = \frac{\sum_{i=1}^{N}(n_i + 1)}{\sum_{i=1}^{N}\left[\frac{(r_{s_{in_i}} - \alpha_{in_i})^2}{r_{s_{in_i}}} + \frac{(\alpha_{i1} - r_{s_{i1}})^2}{r_{s_{i1}}} + \sum_{j=1}^{n_i-1}\frac{(\alpha_{i(j+1)} - \alpha_{ij} + \Delta_{ij})^2}{\Delta_{ij}}\right]}. \tag{29}$$

Hence for the optimal solution to (27), $L_p(\Theta^*, c^*; D_{train}) = L_p(\Theta^*, \gamma(\Theta^*; D_{train}); D_{train})$. In this way, a modified loss function purely designed for the RUL predictor $\varphi(\cdot; \Theta)$ on the training data D_{train} is derived,

$$L_m(\Theta; D_{train}) = L_p(\Theta, \gamma(\Theta; D_{train}); D_{train}). \tag{30}$$

And the optimal value of Θ for $\varphi(\cdot; \Theta)$ comes from the following minimization problem.

$$\Theta^* = \arg\min_{\Theta} L_m(\Theta; D_{train}). \tag{31}$$

(27) is now converted to a normal supervised task for the predictor $\varphi(\cdot; \Theta)$ with the new loss function $L_m(\Theta; D_{train})$. This loss function will be denoted by L_m-loss in later discussions.

From a pure loss analysis view, the loss function (30) introduces prediction-dependent measures at different time-steps between an input sequence and an output sequence with following properties.

1. Strongly controlling at the end-time punishes the end-prediction error.
2. Weakly controlling at the initial-time tolerates the initial prediction

error, and the weakness depends on the sequence length.
3. Controlling the error on prediction increment instead of the direct prediction error at other time-steps punishes the global propagation uncertainty.

3.3. Mini-batch training

In (25), it is revealed that sequential predictions are not independent, such that the joint prediction performance should not simply be measured by the uniform loss for every prediction like the mean square error. Meanwhile in common real-world tasks, not many inspection sequences from independent systems are observed. Under-sampling from limited sequences are essential for data augmentation in predictor training.

However common under-sampling techniques ignore the statistical dependence in the same inspection sequence and also the data imbalance due to variable lengths of inspection sequences. Simple augmentation on the training data leads to over-fitting and imbalance bias when data are not enough for a complicated model.

To keep statistical independence and common rationale in the predictor training, the under-sampling strategy is described as follows.

1. Randomly under-sample a fixed-length sub-sequence from every inspection sequence to form a data batch. In this way, statistical independence and data balance can be promised.
2. Use mini-batch gradient descent algorithms for the loss evaluated on the sampled data batch. Reused data in different batches are not influenced mutually due to the batch isolation and gradient values are updated by the mean.
3. Repeat sampling data batches for the mini-batch gradient descent optimization until optimization converges.

The optimization using mini-batch training for (26) is illustrated in Algorithm 1.

The parameter c for uncertainty propagation modeling interactively influences the predictor design in Algorithm 1, which can also be updated by the validation on the data batch from (29). For practical convenience, the convergence of optimization is substituted in Algorithm 1 by searching the model with the smallest batch-loss when enough training epochs are done.

The algorithm involves three components: batch-based prediction, batch-loss evaluation, and objective-correction. The optimization converges to the optimal values for Θ and c such that the hidden process Y_s in (15) and the RUL predictor $\varphi(\cdot;\Theta)$ in (1) are jointly established.

4. Empirical studies

4.1. Long short-term memory predictor

In previous discussions, the RUL predictor is not specified and the Wiener propagation model is done based on general model assumptions. In this section, the empirical evaluation on a real case study will be presented and the predictor φ is specified as an LSTM network [38].

4.1.1. Network structure

Deep neural networks including CNN, RNN etc., are actively discussed in the literature due to their self-learning capacity for feature representation. Moreover recently RNNs have been widely applied to survival-related prediction [29,39] because of the sequential modeling capacity. LSTM network [30,38] as a special RNN for the RUL prediction in (1) is set up as follows.

Theoretically the predictor φ in (1) can process an inspection sequence with any length. However due to the memory limit and data balance during training, a fixed-length $\eta + 1$, $\eta \in \mathbb{N}$ is always adopted for the uniform input length. From the stationary assumption, time information of subsequences can be removed and an unspecified subsequence since any time is treated as starting from 0, say $\mathbf{x}_{\mathbf{0}:\eta}$.

1. For the sequentially coming input $\mathbf{x}_{\mathbf{0}:\eta}$, the predictor φ iteratively updates the prediction and returns $\alpha_{0:\eta}$ based on two functions σ_h, σ_a for pending parameters $\Theta := \{\theta_0, \theta_h, \theta_a\}$ in the following way.

$$\begin{aligned} &h_0 = \theta_0;\\ &h_i = \sigma_h(h_{i-1}, \mathbf{x_i}; \theta_h);\\ &\alpha_i = \varphi(\mathbf{x}_{\mathbf{0}:\mathbf{i}}; \Theta) = \sigma_a(h_i; \theta_a),\ i = 0, 1, ..., \eta. \end{aligned} \tag{32}$$

2. To be called RNN, the function σ_h in (32) should keep a linear chain rule in general. For instance, with known activation functions σ_h, σ_a and the matrix parameters $\{\theta_j\}_{j=0}^{5}$, the Jordan network [40] as a simple RNN is defined by

$$\begin{aligned} &h_0 = \theta_0;\\ &h_i = \sigma_h(\theta_1\alpha_{i-1} + \theta_2\mathbf{x_i} + \theta_3);\\ &\alpha_i = \sigma_a(\theta_4 h_i + \theta_5),\ i = 0, 1, ..., \eta. \end{aligned} \tag{33}$$

3. Allowing the activation function σ_h in (33) to be replaced by a composite of k activation functions $\{\sigma_k\}_{j=1}^{k}$ leads to different variants of RNN structures, e.g. LSTM and gated recurrent unit (GRU)[30]. Specifically for this modification, parameters in (33) are divided from $(\theta_1, \theta_2, \theta_3)$ in σ_h to $\{(\theta_{1j}, \theta_{2j}, \theta_{3j})\}_{j=1}^{k}$ for corresponding parameters in $\{\sigma_j\}_{j=1}^{k}$.
In this paper the LSTM unit [38] in the following form is specifically

Require:

1. Training data D_{train};
2. Maximum number of training epochs $M \in \mathbb{N}$.

Ensure: A RUL predictor $\varphi(\cdot;\Theta^*)$ and c^* to minimize the loss $L_p(\Theta, c; D_{train})$ in (26).

Initiate $i = 1, c_1, \Theta_1$;
for $i \le M$ **do**
  Randomly under-sample a data batch D_i from D_{train};
  Update $c_{i+1} = \gamma(\Theta_i; D_i)$ from (29) for c;
  Update Θ_{i+1} for Θ by gradient-descent algorithms using the loss function $L_p(\Theta, c_{i+1}; D_i)$;
  $i + 1 \to i$.
end for
$idx = \arg\min_{i=1,...,M} L_p(\Theta_i, c_i; D_i)$;
return $\Theta_{idx} \to \Theta^*$, $c_{idx} \to c^*$.

Algorithm 1. Mini-batch training.

considered,

$$\begin{aligned} h_0 &= \theta_0; \\ f_i &= \sigma_1(\theta_{11}\mathbf{x_i} + \theta_{21}h_{i-1} + \theta_{31}); \\ g_i &= \sigma_2(\theta_{12}\mathbf{x_i} + \theta_{22}h_{i-1} + \theta_{32}); \\ o_i &= \sigma_3(\theta_{13}\mathbf{x_i} + \theta_{23}h_{i-1} + \theta_{33}); \\ e_i &= f_i \circ e_{i-1} + g_i \circ \sigma_4(\theta_{14}\mathbf{x_i} + \theta_{24}h_{i-1} + \theta_{34}); \\ h_i &= o_i \circ \sigma_5(e_i); \\ \alpha_i &= \sigma_a(\theta_4 h_i + \theta_5), \quad i = 0, 1, ..., \eta, \end{aligned} \tag{34}$$

where the operator $\circ$ denotes the Hadamard product. Moreover, σ_1, σ_2, σ_3 are sigmoid function and σ_4, σ_5 are hyperbolic tangent function [38]. σ_a related to the output will be specified as ReLu [41] in later discussions. Other choices of activation functions are also possible [42].

4. From (34), $\{\theta_0, \{(\theta_{1j}, \theta_{2j}, \theta_{3j})\}_{j=1}^4, \theta_4, \theta_5\}$ forms all trainable parameters Θ for the LSTM prediction $\varphi(\cdot; \Theta)$. And the output sequence $\alpha_{0:\eta}$ forms the LSTM output.

For more discussions about RNN structures, readers can refer to [30].

RUL prediction is a special task with only non-negative observations. To promise the non-negativeness of RUL prediction, the output layer is set as a time-distributed dense layer with ReLu activation [41], where ReLu is given by $\max(0, x)$ with input x. This simply replaces the negative prediction with 0.

Later empirical tests on the LSTM predictor will be done in a laptop workstation with the high-level framework Keras [43] and the Tensorflow back-end [44].

4.2. Turbofan engine degradation simulation data

4.2.1. Available data

Provided by NASA using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), the turbofan engine degradation data for run-to-failure simulations [4,45] will be considered for empirical evaluation in this paper.[2]

The degradation data come from the simulation of a turbofan engine (illustrated in Fig. 1) in the C-MAPSS platform. In the NASA repository 4 data-sets $FD001 - 004$ are presented, and a fifth C-MAPSS data-set is made public in the PHM'08 data challenge [4]. Due to the high-variability and high-dimensionality, these data are well-known benchmarks in recent years to test RUL predictors in the high-dimensional scenario, and a good summary regarding recent studies on this data-set is presented in [45].

In later experiments, data-sets labelled by *FD*004 will be considered.

- Every inspection returns a 24-dimensional vector consisting of 21 sensor readings and 3 operation settings.
- 249 run-to-failure sequences exist for training, consisting of 61,249 different time-stamped inspection data. The maximal and minimal number of inspection data in all sequences are 543 and 128 respectively.
- 248 randomly truncated sequences exist for testing, labelled by actual RUL records at truncating times. The maximal and minimal number of inspection data in all sequences are 486 and 19 respectively.
- Sequences from different run-to-failure tests have different lengths.

It is assumed that all engines are identical and independent such that data from different engines are independent.

[2] Used data-sets are publicly available at the web site of NASA (https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan).

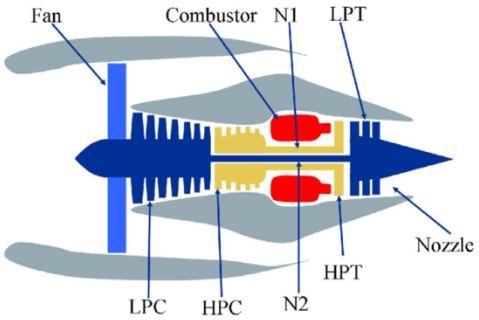


Fig. 1. Diagram of engine simulated in C-MAPSS (Fig. 1 in [4]).

4.2.2. Setup for LSTM predictor

As described in (34), the LSTM predictor is set up as follows.

- The LSTM predictor has 127 time steps, 256 hidden units and 288,001 trainable parameters. Each time step accepts an inspection vector of 24 dimensions.
- The LSTM has a mask input layer for input sequences of shorter lengths than 127.
- The LSTM predictor has time distributed outputs of length 127.
- The RMSprop optimizer with initial learning rate 0.01 is adopted.
- To compare training results from the L_m-loss derived in (30), the mean square error (MSE) with equal weights to prediction errors at different time steps is considered.

4.2.3. Batch data preparation

In a rigorous setting, existing 249 training sequences are not enough for the LSTM predictor even each sequence consists of hundreds of sensor readings.

- All inspection data in the training sequences and testing are pre-processed by the same scaling and normalization.
- Training data batches consist of 249 sub-sequences of length 127 from different training sequences, which are associated with the actual RUL value at the end-time in the sub-sequence.
- The testing data batch consists of 248 sub-sequences of length 127 ending at truncating times from different testing sequences, which are associated with the actual RUL value at the end-time in the sub-sequence. The testing data are denoted by D_{test}.
- Due to random data batch selection, the training epoch is set to include 250 without actual meaning.
- The predictor training will be done for 100 epochs under different loss functions, and the model reaching the minimal batch loss during training is selected for later testing.

4.3. Results and discussions

During training, the LSTM predictor only accepts a 127 time-step inspection sequence, and the prediction can slide over calendar time if the available inspection sequence is longer than 127 time-steps. As the length is not a network parameter, the trained network weights can be copied to another LSTM with longer time-steps such that the complete sequence in the testing data can be evaluated. Moreover the prediction models trained by the L_m-loss and MSE are called Wiener-hidden model and the MSE-oriented model respectively in this section.

4.3.1. Training performance

Due to the random batch preparation strategy, no static training data are prepared to show the training performance over time. However the predictor can still be roughly evaluated on the randomly sampled data batch, whose losses over training epochs are illustrated in Fig. 2.

From Fig. 2, the Wiener-hidden model converges much quicker than

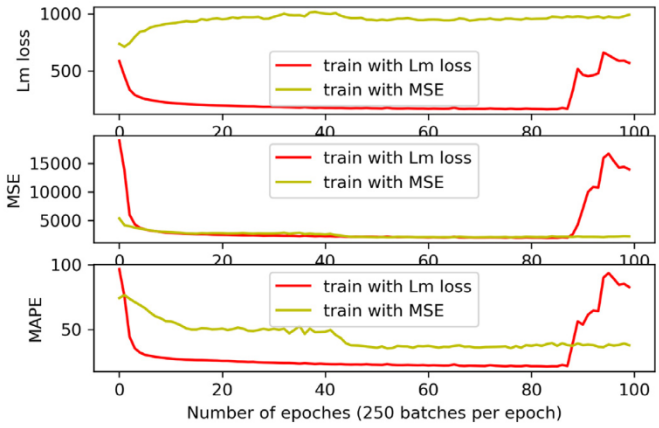


Fig. 2. Different losses over training epoches: L_m-loss in (30), MSE and MAPE.

the MSE-oriented model for the L_m-loss and MAPE, while the MSE-oriented model converges quicker if MSE loss is considered. Moreover during 100 training epochs, the optimal MSE-oriented model has a significant difference with the optimal Wiener-hidden model for the L_m-loss and MAPE.

4.3.2. Point-wise testing performance

For the *sequence* − *RUL* pair in the testing data, the point-wise prediction performance can be evaluated by the last output from the LSTM predictor. Comparing the Wiener-hidden prediction with the MSE-oriented prediction, the necessity of introducing the Wiener propagation model is illustrated in Fig. 3. The Wiener-hidden prediction shows more accuracy at a near-failure time, but its prediction accuracy at a far-to-failure time is sacrificed. Hence by introducing the Wiener propagation model, predictions over calendar time are well controlled and the prediction uncertainty over calendar time is redistributed.

As the training data come from run-to-failure tests, fewer data exist at the far-to-failure time. Hence the prediction is naturally under-trained at a far-to-failure time due to lack of data. This also explains why the sample variance visually increases over the actual RUL.

As some testing sequences are shorter than the length of LSTM predictor, so predictions with less input data show an exceptional increased uncertainty near failure times in Fig. 3.

At the end, the LSTM output at the truncating time on the testing data is evaluated by metrics as in Table 1. In Table 1, 5 different metrics are considered: MSE, mean absolute error(MAE), MAPE, the uncertainty propagation rate estimated from (29), and the L_m-loss defined in (30). It is remarked that these metrics measured the point-wise

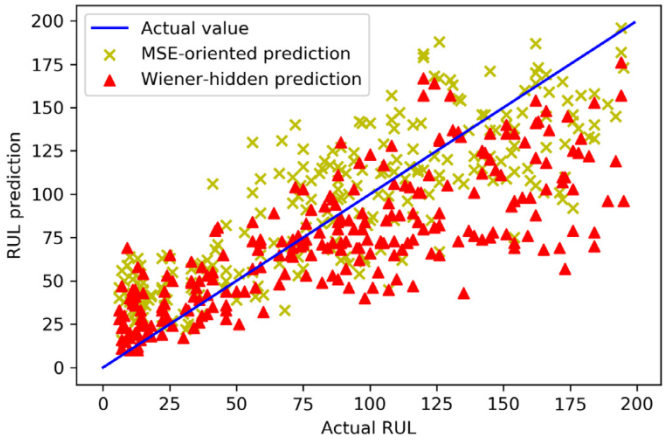


Fig. 3. Actual v.s. predicted RUL at the truncating time for the testing data batch.

Table 1
The LSTM output at the truncating time on the testing data under different losses. Θ^* represents the LSTM predictor's parameters, and $\gamma(\Theta^*; D_{test})$ represents the estimate of the uncertainty propagation rate c by (29).

Loss metric	MSE prediction	Wiener prediction
MSE	**942.22**	1354.21
MAE	**25.25**	27.50
MAPE	**70.08**	53.93
$\gamma(\Theta^*; D_{test})$	0.0116	**2.98**
$L_m(\Theta^*; D_{test})$	799.47	**199.31**

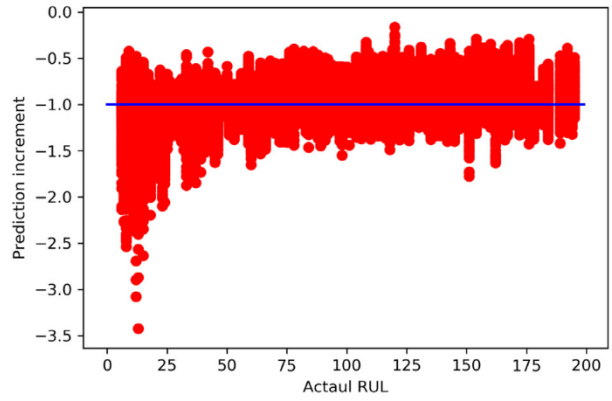
performance of the LSTM predictor, while the sequence-level prediction uncertainty is not actually considered. This is a trade-off to allow a uniform-length input sequence.

From a pure accuracy view, the MSE-oriented prediction without punishment over time as in the MSE and MAE outperforms the Wiener-hidden prediction. However when time-related punishment is introduced as in MAPE and L_m-loss, the Wiener-hidden prediction outperforms the MSE-oriented prediction.

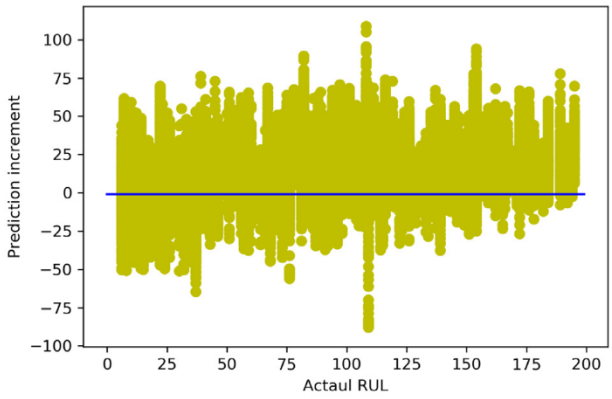
4.3.3. Propagation analysis for sequential RUL prediction

The testing performance is evaluated in the last sub-section in a point-wise way, this sub-section continues to test the sequence-level prediction on the testing data.

The increments of LSTM output sequences for the MSE-oriented model and Wiener-hidden model are illustrated in Fig. 4. From the comparison, the Wiener-hidden prediction increments visually show stationary, independent, and Gaussian distributed properties ($\sim \mathcal{N}(0, 1)$) as modeled in the Wiener propagation model, while the

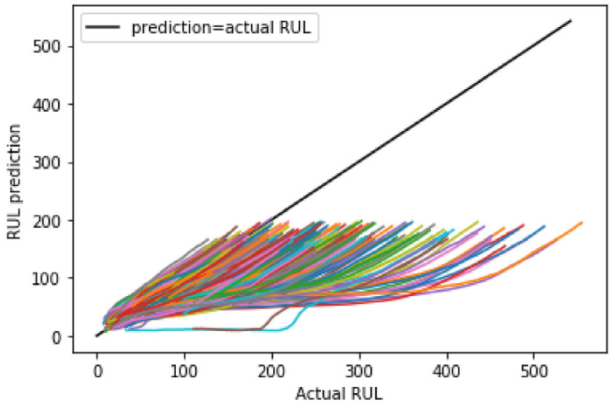


(a) Wiener-hidden prediction

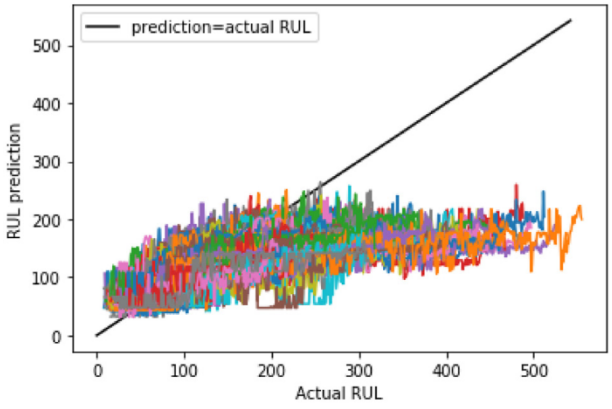


(b) MSE-oriented prediction

Fig. 4. Increments at neighbored time-steps for the Wiener-hidden and MSE-oriented prediction over the actual RUL for the testing data batch.



(a) Wiener-hidden prediction



(b) MSE-oriented prediction

Fig. 5. Prediction sequences over the actual RUL for the testing data, respectively from the Wiener-hidden and MSE-oriented models.

MSE-oriented prediction increments lose control and shows a large range (− 50 to 75). It is also noted that the testing sequences are truncated within 200 time-steps to failures, which are located in the best performance period from Fig. 3.

Moreover the prediction sequences for all testing sequences are completely produced in Fig. 5 for the Wiener-hidden model and the MSE-oriented model. It is interesting to observe that the sequence-level variety is comparable with the linear propagated variance of Wiener process, where the increasing trend is significant. Also compared with the MSE-oriented prediction, the Wiener-hidden predictions are visually more ”peaky” at near-failure times and more ”wide-spreaded” at far-to-failure times. Hence we concludes that the Wiener propagation model improves the near-failure prediction accuracy by sacrificing the far-to-failure prediction accuracy.

5. Conclusions

To control the trade-off between accuracy and uncertainty for the sequential RUL prediction in a high-dimensional inspection scenario, this paper provided a combined framework for machine learning RUL predictors and stochastic process modeling. This allows us to make an explicit trade-off between prediction performance and knowledge about the uncertainty of the predictions. Compared with existing techniques for RUL prediction, conclusive remarks are given as follows.

- The problem setup in this paper is moved from conventional point-wise RUL prediction to a new scope of sequential prediction. Hence instead of considering the “average” prediction performance, the path-dependent “bias” over prediction sequences is introduced and modeled by the drifted Wiener process. Our work shows a great potential to be integrated into predictive maintenance, where real-time maintenance decisions are made sequentially and sequential prediction is more applicable.
- The methodology of surrogate modeling to process prediction uncertainty instead of observation uncertainty indirectly solves the difficult modeling problem for high-dimensional inspection data. An uncertainty propagation modeling framework is newly proposed for data-driven RUL predictors, moreover interactive training between the surrogate model and the RUL predictor is proposed and realized.
- The near-failure prediction accuracy can be improved by relaxing the control for far-to-failure prediction accuracy, which is verified from the empirical results in a case study and also the theoretical uncertainty control of uncertainty propagation model.

Declaration of Competing Interest

There is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review.

Acknowledgements

This work was supported by European Union’s Horizon 2020 grant no.766994 (PROPHESY). Y. Deng also thanks for the support from National Natural Science Foundation of China (NSFC) grant no.71701143.

References

[1] Si X-S, Wang W, Hu C-H, Zhou D-H. Remaining useful life estimation - a review on the statistical data driven approaches. Eur J Oper Res 2011;213(1):1–14.
[2] Yam RCM, Tse PW, Li L, Tu P. Intelligent predictive decision support system for condition-based maintenance. Int J Adv Manuf Technol 2001;17(5):383–91.
[3] Rengasamy D, Morvan HP, Figueredo GP. Deep learning approaches to aircraft maintenance, repair and overhaul: a review. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). 2018. p. 150–6. https://doi.org/10.1109/ITSC.2018.8569502.
[4] Saxena A, Kai G, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. International Conference on Prognostics and Health Management. 2008. p. 1–9.
[5] Elforjani M, Shanbr S. Prognosis of bearing acoustic emission signals using supervised machine learning. IEEE Trans Ind Electron 2018;65(7):5864–71. https://doi.org/10.1109/TIE.2017.2767551.
[6] Jardine AKS, Lin D, Banjevic D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mech Syst Signal Process 2006;20(7):1483–510.
[7] Pandey MD, Yuan XX, van Noortwijk JM. The influence of temporal uncertainty of deterioration on life-cycle management of structures. Struct Infrastruct Eng 2009;5(2):145–56. https://doi.org/10.1080/15732470601012154.
[8] Tian Z, Liao H. Condition based maintenance optimization for multi-component systems using proportional hazards model. Reliab Eng Syst Saf 2011;96(5):581–9. https://doi.org/10.1016/j.ress.2010.12.023. http://linkinghub.elsevier.com/retrieve/pii/S0951832010002796
[9] Cox DR. Regression models and life-tables. J R Stat Soc 1972;34(2):187–220.
[10] Gebraeel N, Lawley M, Liu R, Parmeshwaran V. Residual life predictions from vibration-based degradation signals: a Neural network approach. Residual life predictions from vibration-based degradation signals: a neural network approach 2004;51(3):694–700. https://doi.org/10.1109/TIE.2004.824875. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1302346.
[11] Tian Z. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. J Intell Manuf 2012;23(2):227–37.
[12] Tian Z, Wong L, Safaei N. A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. Mech Syst Signal Process 2010;24(5):1542–55.
[13] Mahamad AK, Saon S, Hiyama T. Predicting remaining useful life of rotating machinery based artificial neural network. Comput Math Appl 2010;60(4):1078–87.
[14] Huang R, Xi L, Li X, Richard Liu C, Qiu H, Lee J. Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. Mech Syst Signal Process 2007;21(1):193–207. https://doi.org/10.1016/j.ymssp.2005.11.008. 0208024
[15] Zhang Y, Ding X, Liu Y, Griffin P. An artificial neural network approach to transformer fault diagnosis. IEEE Transactions on Power Delivery 1996;11(4):1836–41.
[16] Guo L, Li N, Jia F, Lei Y, Lin J. A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 2017;240:98–109. https://doi.org/10.1016/j.neucom.2017.02.045.

[17] Heimes F. Recurrent neural networks for remaining useful life estimation. Prognostics and Health Management, 2008 PHM 2008 International Conference on 2008:1–6. https://doi.org/10.1109/PHM.2008.4711422. http://ieeexplore.ieee.org.etechconricyt.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=4711422
[18] Liu J, Saxena A, Goebel K, Saha B, Wang W. An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. Annual Conference of the Prognostistics and Health Mangement Society, PHM. 2010.
[19] Zhu J, Chen N, Peng W. Estimation of bearing remaining useful life based on multiscale convolutional neural network. IEEE Trans Ind Electron 2019;66(4):3208–16. https://doi.org/10.1109/TIE.2018.2844856.
[20] Babu GS, Zhao P, Li X-L. Deep convolutional neural network based regression approach for estimation of remaining useful life. International conference on database systems for advanced applications. Springer; 2016. p. 214–28.
[21] Le Cun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.
[22] Koziel S, Ciaurri DE, Leifsson L. Surrogate-based methods. In: Koziel S, Yang XS, editors. Computational Optimization, Methods and Algorithms, Stud Comput Intell 356. 2011. https://doi.org/10.1007/978-3-642-20859-1_3.
[23] Zaytsev A, Burnaev E. Large scale variable fidelity surrogate modeling. Ann Math Artif Intell 2017;81:167–86.
[24] Garnelo M, Rosenbaum D, Maddison CJ, Ramalho T, Saxton D, Shanahan M, et al. Conditional neural processes. International conference on machine learning. 2018. p. 1690–9.
[25] Ting Lee M-L, DeGruttola V, Schoenfeld D. A model for markers and latent health status. J R Stat Soc 2000;62(4):747–62. https://doi.org/10.1111/1467-9868.00261.
[26] Bengio Y, De Mori R, Flammia G, Kompe R. Global optimization of a neural network-hidden Markov model hybrid. IEEE Trans Neural Netw 1992;3(2):252–9. https://doi.org/10.1109/72.125866.
[27] van Noortwijk J. A survey of the application of Gamma processes in maintenance. Reliab Eng Syst Saf 2009;94(1):2–21.
[28] Cai B, Huang L, Xie M. Bayesian networks in fault diagnosis. IEEE Trans Ind Inf 2017;13(5):2227–40. https://doi.org/10.1109/TII.2017.2695583. 1–1
[29] Katzman JL, Shaham U, Cloninger A, Bates J, Jiang T, Kluger Y. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. BMC Med Res Methodol 2018;18(1):24. https://doi.org/10.1186/s12874-018-0482-1.
[30] Graves A. Supervised sequence labelling with recurrent neural networks. Springer Berlin Heidelberg; 2012.
[31] Si X-S, Wang W, Hu C-H, Zhou D-H. Estimating remaining useful life with three-source variability in degradation modeling. Reliab IEEE Trans 2014;63(1):167–90.
[32] Liao H, Elsayed EA, Chan L-Y. Maintenance of continuously monitored degrading systems. Eur J Oper Res 2006;175(2):821–35.
[33] Ye Z-S, Chen N. The inverse gaussian process as a degradation model. Technometrics 2014;56(3):302–11.
[34] Jeanblanc M, Yor M, Chesney M. Mathematical methods for financial markets. Springer Science & Business Media; 2009.
[35] Bérenguer C, Grall A, Dieulle L, Roussignol M. Maintenance policy for a continuously monitored deteriorating system. Probab Eng InfSci 2003;17(2):235–50.
[36] Deng Y, Barros A, Grall A. Degradation modeling based on a time-dependent ornstein-uhlenbeck process and residual useful lifetime estimation. Reliab IEEE Trans 2016;65(1):126–40.
[37] Chao MT. Degradation analysis and related topics: some thoughts and a review. Proc Natl Sci Counc Repub China, Part A Phys Sci Eng 1999;23(5):555–66.
[38] Gers FA, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Comput 1999;12:2451–71.
[39] Zhu X, Yao J, Zhu F, Huang J. Wsisa: Making survival prediction from whole slide histopathological images. IEEE Conference on Computer Vision and Pattern Recognition. 2017. p. 6855–63.
[40] Jordan MI. Chapter 25 - serial order: a parallel distributed processing approach. In: Donahoe JW, Dorsel VP, editors. Neural-Network Models of Cognition Advances in Psychology, 121. North-Holland; 1997. p. 471–95. https://doi.org/10.1016/S0166-4115(97)80111-2. http://www.sciencedirect.com/science/article/pii/S0166411597801112.
[41] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. Proceedings of ICML 2010. 2010. p. 807–14. http://dl.acm.org/citation.cfm?id=3104322.3104425.
[42] Gers FA, Schraudolph NN, Schmidhuber J. Learning precise timing with LSTM recurrent networks. JMachLearnRes 2002;3(Aug):115–43.
[43] Chollet F., et al. Keras. 2015. https://keras.io.
[44] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., et al. TensorFlow: large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org; https://www.tensorflow.org/.
[45] Ramasso E, Saxena A. Performance benchmarking and analysis of prognostic methods for CMAPSS datasets. Int J Prognost Health Manage 2014;5:1–15.