Semi-supervised Learning with Deep Generative Models for Asset Failure Prediction

Andre S. Yoon, Taehoon Lee, Yongsub Lim, Deokwoo Jung
Philgyun Kang, Dongwon Kim, Keuntae Park, Yongjin Choi
Big Data Tech. Lab
SK Telecom

ABSTRACT

This work presents a novel semi-supervised learning approach for data-driven modeling of asset failures when health status is only partially known in historical data. We combine a generative model parameterized by deep neural networks with non-linear embedding technique. It allows us to build prognostic models with the limited amount of health status information for the precise prediction of future asset reliability. The proposed method is evaluated on a publicly available dataset for remaining useful life (RUL) estimation, which shows significant improvement even when a fraction of the data with known health status is as sparse as 1% of the total. Our study suggests that the non-linear embedding based on a deep generative model can efficiently regularize a complex model with deep architectures while achieving high prediction accuracy that is far less sensitive to the availability of health status information.

Keywords

Data-driven prognostics; semi-supervised learning; deep learning; deep generative models; predictive maintenance

1. INTRODUCTION

With increasingly more sensor data available either transient in stream or permanent in database, numerous physical assets in different industrial sectors such as manufacturing, transportation, and energy are subjected to asset condition and health monitoring [17]. When sufficient information present in the data, referred to as condition monitoring (CM) data, the asset health status and its evolution can be modeled to predict a specific point in time after which an asset will no longer meet its normal operational requirements, i.e., asset failure [16]. The concept of RUL, also referred to as an estimated time to failure, is widely used in the future reliability prediction with an important industrial application known as predictive maintenance (PdM) [21]. The PdM attempts to reduce costly unscheduled maintenance or maintenance that is excessively scheduled.

Existing approaches for predictive maintenance belong two categories: physics-driven (deduction from physical equations underlying mechanical operations) and data-driven (induction from measurements under actual operations). In this paper, we narrow down our scope to only focus on data-driven methods. Gaussian and Markov processes have produced promising results [16, 28, 29] in the field of predictive maintenance since 1990's, but they often suffer from practical limitations such as scalability (*i.e.*, a basic form of Gaussian process requires $O(n^3)$). On small datasets, they are very useful because strong priors can cover lack of data with reasonable computations. However, as data volumes are exploding, more complicated functions are required to model massive, high-dimensional operational measurements which are hard to be characterized by well-known priors and kernel functions. The advent of deep learning has enabled us to capture the complicated functions; for example, [31] exhibited performance improvements with deep learning-based prognostic models to predict the RUL of given assets.

While many deep learning-based approaches rely on the abundance of historical data with the availability of corresponding health indicators either from the CM data itself or from extra measurements or modelings, obtaining exact health status of a given asset in reality is highly expensive or impossible in some cases. It is because labeling requires expertise or special devices to produce it, and generally highly time consuming. Above all, failures are rare and not easily replicated, especially for those that are mission-critical. This is also closely related to the problem of intermittent failures also known as No Fault Found (NFF) problem [16]. For example, in the case of mission-critical assets such as an aircraft engine or equipment in nuclear power plant, even a single failure is catastrophic and therefore, regular maintenance or replacement is carried out regardless of the true condition of an asset. Or in the case of fresh start of newly installed assets, one may have to wait long to collect any run-to-failure data to precisely model the degradation and consequently failure.

In the lack of sufficient health status information, which we call a label or a target variable, y, data-driven modeling of degradation becomes immensely difficult in many cases. It is often the case where there is abundant data with relatively few labels since unlabeled data is relatively cheap. We recognize while the complexity of various data-driven models increases, especially with the latest deep neural net architectures, there has been relatively little attention paid to the problem of insufficient labels in data-driven prognostics. In the field of machine learning, researchers have long

recognized such difficulty and explored various techniques to make learning better generalized in the lack of sufficient labels [3].

Semi-supervised learning (SSL) is a class of techniques that falls between unsupervised learning and supervised learning (SL) [3]. While supervised learning relies solely on labeled data, SSL makes use of unlabeled data as well to improve learning accuracy. Thus, SSL is resorted to deal with a situation where relatively few labeled data are available with large unlabeled data. There exist many different approaches in SSL, which include heuristic approaches such as self-learning and approaches based on generative models, and low-density separation [3]. Among different approaches, generative models parameterized by deep neural networks referred to as deep generative models (DGM) have recently achieved state-of-the-art performance in various unsupervised and semi-supervised learning tasks [14, 19, 25]. Two most popular approaches of DGM rely on variational inference [15] and generative adversarial network (GAN) [6]. The latter being probably the most popular in deep learning research nowadays is grounded in game theory.

In this paper, we propose a semi-supervised learning technique to improve data-driven asset failure prediction when labeled data is highly limited compared to the unlabeled. For example, we consider the case where historical CM data are abundant from a fleet of assets but it is only partially labeled with health status. In such a case, we observe the prediction accuracy becomes highly degraded even with sophisticated deep learning models based on the Recurrent Neural Networks (RNN) architectures such as Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM). Our proposed method uses a non-linear embedding, which is obtained from one of the DGMs known as variational autoencoder (VAE). The variational autoencoder makes use of all the available data, both labeled and unlabeled. A part of the VAE is taken to form an embedding network, which is used to define a latent space where a model for reliability prediction is trained.

In the remainder of this paper, we first give the background of this work in Section 2 by discussing deep neural networks and a variational autoencoder as a choice of deep generative model. We describe our proposed methods in Section 3. In Section 4, we further the details of our methods with the experimental evaluations. Lastly, we discuss our findings in Section 5.

2. BACKGROUND

2.1 Problem Statement

In our problem, we have multivariate data for sensor measurement and their corresponding RUL. Let $x^{(i,j)}$ denote i_{th} measurement value of sensor j. Furthermore, let us $\mathbf{x^{(i)}}$ denote a vector of multivariate sensor measurement such that $\mathbf{x^{(i)}} = [x^{(i,1)}, \cdots, x^{(i,m)}]$ where m is the number of sensors. Formally, we describe a sensor measurement matrix by denoting $\mathbf{x} = \{\mathbf{x^{(1)}}, \cdots, \mathbf{x^{(n)}}\}$ where $\mathbf{x^{(i)}} \in \mathbb{R}^m$ and n is the number of observed measurements. We also denote the corresponding set of RUL by $\mathbf{y} = \{y^{(1)}, \cdots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}^+$. More compactly, we define the data set $\mathcal{D}$ by $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1,\cdots,n}$.

In general, the data-driven prognostics approach is to learn the best predictor of RUL (or equivalent health indicator used to predict RUL) from the previously observed data

set $\mathcal{D}^T$, i.e. training data set. The challenging problem is to find an appropriate transformation of the raw measurement $\mathbf{X}$ to efficiently learn a reliable RUL predictor. Such problem can be described by finding a mapping function F such that $F : \mathbf{x} \mapsto \mathbf{z}$ where a latent variable $\mathbf{z} \in \mathbb{R}^k$ and $m < k$. Then the optimal predictor can be defined by a function of $\mathbf{z}$ with parameter γ denoted by $f_\gamma(\mathbf{z})$ such that $f_\gamma(\mathbf{z}) = \underset{y}{argmax}\ p(y \mid \mathbf{z}, \gamma)$.

In traditional machine learning approaches, it first finds appropriate latent variable $\mathbf{z}$ by well known linear transformation methods (e.g. Principal Component Analysis) often with heuristics. Then it estimates the optimal parameter γ by training $f_\gamma(\mathbf{z})$ with $\mathcal{D}^T$ given a certain simplified assumption for the distribution $p(y \mid \mathbf{z}, \gamma)$(e.g. multivariate normal distribution). Such conventional approaches inherently has its own limitations in providing a general solution for RUL prediction. First, it is often that the mapping function F for latent variable $\mathbf{z}$ is non-linear and its specific form is largely unknown. Second, any strong assumption on distribution $p(y \mid \mathbf{z}, \gamma)$ often poorly captures complex cross dependency over time and sensors. Meanwhile, deep neural network can provide a generic solution to jointly learn the non-linear mapping function F and the predictor f_γ from training data set $\mathcal{D}^T$ using directed graph network with various levels of abstraction. We drop the superscript T for the rest of the paper.

In our problem setting, we assume that only a small fraction of the samples has label information. Let $\mathcal{D}_L^T$ and $\mathcal{D}_U^T$ denote the part of the data set with labels and the part without the labels, respectively. The primary goal in this paper is to develop deep learning-based approach to predict RUL from $\mathcal{D}_L^T$ where $|\mathcal{D}_L^T| \ll |\mathcal{D}_U^T|$, which is a class of semi-supervised learning. In the semi-supervised learning setting, we assume that $\mathcal{D}_L^T$ comes from exactly the same distribution of $\mathcal{D}_U^T$. Our semi-supervised approach fully exploits $\mathcal{D}_U^T$ to learn the non-linear mapping function f to get a robust feature representation of $\mathbf{x}$ for stable RUL predictor f_γ given $\mathcal{D}_L^T$.

2.2 Neural Network Overview

In this section we briefly discuss the definition and the architectures of basic deep neural network and its variants.

2.2.1 feed-forward neural network

We can formally describe a feed-forward neural network as follows. Let $f_{l,i}$ and $g_{l,i}$ denote a linear pre-activation and non-linear activation function of ith neuron at layer l, respectively where $f_{l,i}, g_{l,i} \in \mathbb{R}^k$ for $i = 1, \cdots, n_l$, n_l is the number of neurons at layer l, and k is the dimension of a neuron's outputs. We can conveniently assume that the output dimension k is the dimension of a single sensor measurement. In our problem setting, $k = 1$ since each sensor measurement is a scalar value. Then layer l has two function vectors ; $f_l = [f_{l,1} \cdots f_{l,n_l}]^T$ and $g_l = [g_{l,1} \cdots g_{l,n_l}]^T$.

Let $\mathbf{x}_l = [\mathbf{x}_{l,i}]_{i=1,\cdots,n_l}$ denote activation output at layer l where $\mathbf{x}_{l,i} = g_{l,i} \circ f_{l,i}(\mathbf{x}_{l-1})$. The pre-activation of layer l is computed by linear transformation

$$\mathbf{f}_l(\mathbf{x}_{l-1}) = \mathbf{W}_l \mathbf{x}_{l-1} + \mathbf{b}_l \qquad (1)$$

where $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{n_{l-1}}$.

For the non-linear activation, we use a popular rectified linear unit (ReLU) [22] where $g_i(f_i) = \max\{0, f_i\}$ for $f_i \in \mathbb{R}$. The layer l's parameter, $\theta_l = (\mathbf{W}_l, \mathbf{b}_l)$ is estimated by

the *backpropagation algorithm* [18] given $\mathcal{D}^T$. Finally, feed-forward neural network of L layers is defined as following :

$$F(\mathbf{x}; \theta) = f_{out} \circ h_L(\mathbf{x}) \qquad (2)$$

where $h_l = (g_l \circ f_l) \circ h_{l-1}$ and $\theta = (\theta_0, \cdots, \theta_L)$.

2.2.2 Recurrent Neural Networks

The feed-forward neural network we described in previous section does not allow temporal dependency of samples to be embedded in its model. In our problem, the current sample are mostly likely to be dependent on its previous samples. Hence, we need a more generalized model capable of elegantly incorporating temporal dependency in it. RNN is such kind of a model that can capture into model the influence of an arbitrary number of previous samples on the current [5]. The basic RNN structure consists of a cell with a cyclic loop whose internal state evolves over time by the current sample input and its previous state output. The output of RNN is simply a series of cell states. When we represents the cell state at each time as a neuron, its resultant structure is a non-cyclic horizontally connected neural network where the current neuron's output feeds into the next neuron's input.

Using the horizontal representation of cell states we can formally describe RNN with a simple modification of (1) and (2). Let us assume that we use the previous T samples for RNN input. Then the RNN input at time t is the array of multivariate sensor measurements from t to t_T denoted by $\mathbf{x}_{t-T}^t = [\mathbf{x}^t, \cdots, \mathbf{x}^{t-T}]$. Let $\mathbf{s}_t$ denote a cell state at time t such that $\mathbf{s}_t \in \mathbb{R}^k$ where k is a dimension of the state. Then pre-activation function of a neuron is defined by a linear mapping of the cell's previous state $\mathbf{s}_{t-1}$ and the current input $\mathbf{x}_t \in \mathbb{R}^m$ at time t as shown in (3).

$$f(\mathbf{s}_{t-1}, \mathbf{x}_t) = \mathbf{U}\mathbf{s}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b} \qquad (3)$$

where $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{U} \in \mathbb{R}^{k \times k}$, and $\mathbf{W} \in \mathbb{R}^{k \times m}$.

The cell state at time t is updated by a non-linear activation function g on the f output such that $\mathbf{s}_t = g \circ f(\mathbf{s}_{t-1}, \mathbf{x}_t) = h(\mathbf{s}_{t-1}, \mathbf{x}_t)$. Finally, RNN for T_{th} order time dependency is compactly defined as following :

$$F(\mathbf{x}_{t-T}^t; \theta) = [h(\mathbf{s}_{t'-1}, \mathbf{x}_{t'})]_{t'=t, \cdots, t-T} \qquad (4)$$

where $h = (g \circ f)$ and $\theta = (\mathbf{U}, \mathbf{W}, \mathbf{b})$. Note that all neural cells share parameter θ over time.

The RNN parameter θ can be trained using backpropagation through time (BPTT) algorithm [30]. However, RNNs with the BPTT suffer from the *vanishing* and *exploding* gradient problem [2] because a recurrent network grows as deep as sequence length. One of popular solutions is to exploit Long Short-Term Memory (LSTM). An LSTM [9] is a second-order recurrent network that has three additional gate layers called *input*, *forget*, and *update* gates. The gate parameters helps a recurrent network have good information flow over time by alleviating the vanishing gradient. Thus, a recurrent network *with* gates can work better than one *without* gates as more long-term dependencies exist.

GRU [4] is a simpler variant of LSTM which combines *forget input* gate and *read input* gate into *overwrite* gate by setting an input gate to the inverse of forget gate, ,i.e $i_t = 1 - f_t$. It is shown that the performance of GRU is on par with LSTM but more computationally efficient [10].

In our study, we find there is no meaningful difference in prediction accuracy between the two variants.

2.3 Variational Autoencoder (VAE)

An autoencoder is a neural net with an encoder-decoder architecture that is trained to reconstruct its own input. Given an input $\mathbf{x}$, the autoencoder, $f_{enc}(\mathbf{x})$, encodes and then the decoder, $f_{dec}(\mathbf{z})$, decodes the encoded input (a *representation*) back to reconstruct the input as closely as possible by minimizing a loss function of the form [5]:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - f_{dec}(f_{enc}(\mathbf{x}))|| \qquad (5)$$

where f_{enc} and f_{dec} are symmetric in topology. The autoencoder is trained using backpropagation algorithm in the same manner as in the feed-forward neural network in Section 2.2.1. However, when the architecture becomes deep with many hidden layers, pretraining is necessary to set the initial weights of the autoencoder close to the final ones [8]. The VAE inherits autoencoder's encoder-decoder architecture, but it is different in that the latent representation $\mathbf{z}$ of given data $\mathbf{x}$ are replaced with stochastic variables. The encoder and the decoder of VAE are probabilistic and given by $q_\phi(\mathbf{z}|\mathbf{x})$, an approximate posterior, and $p_\theta(\mathbf{x}|\mathbf{z})$, likelihood of the data $\mathbf{x}$ given the latent variable $\mathbf{z}$, respectively. The approximate posterior is parameterized by a neural network and the likelihood is given by a multivariate Gaussian whose probabilities are obtained from $\mathbf{z}$.

The objective function of VAE is the variational lower bound to the marginal likelihood of the data, i.e., $\log p_\theta(\mathbf{x}) = \sum_{i=1}^n \log p_\theta(\mathbf{x^{(i)}})$. It can be rewritten for individual data points, $\mathbf{x}^{(i)}$ as [15]:

$$\log p_\theta(\mathbf{x}^{(i)}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \quad (6)$$

where D_{KL} is Kullback-Leibeler (KL) divergence. Since $D_{KL} \geq 0$, (6) can be rewritten as:

$$\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \qquad (7)$$

The left-hand side of (7), which is called *variational lower-bound*, can further be rewritten using Bayes' rule and the definition of KL divergence as [15]:

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\big] \quad (8)$$

which can be viewed as the sum of reconstruction error plus a term that acts as a regularization.

While obtaining the gradients of the encoder is relatively straightforward, that of the decoder is not. This is because, the reconstruction error term in Eq. 8 requires the Monte Carlo estimate of the expectation, which is not easily differentiable [15]. This problem is solved by introducing a reparameterization of z with a *deterministic variable* such that $z = \mu + \sigma\epsilon$, with $\epsilon \sim \mathcal{N}(0, 1)$, which is known as "reparameterization trick" [15]. With this trick, a differentiable estimator of the variational lower bound can be obtained and the backpropagation algorithm can be applied to train VAE. The overall architecture of VAE is shown in Fig. 2.

3. OUR APPROACH

3.1 Overview

In this section, we describe our overall approach for semi-supervised learning for RUL prediction using neural network framework and a deep generative model described in
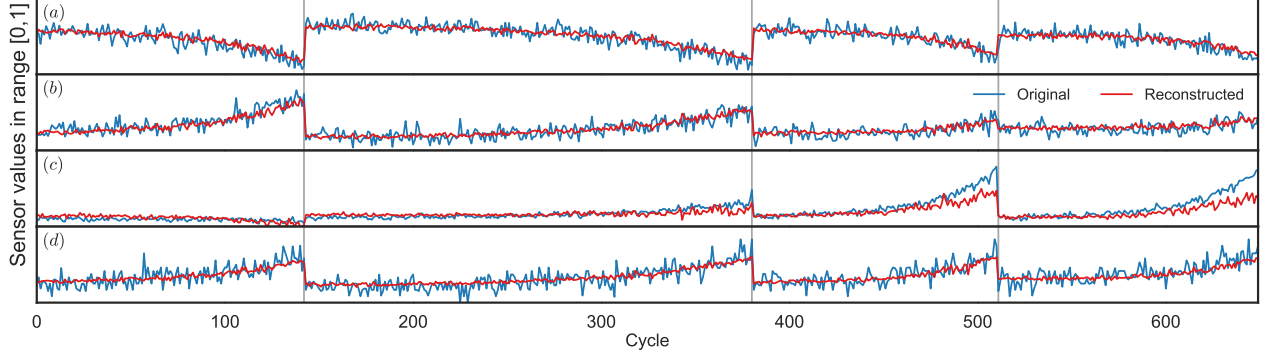
Figure 1: Normalized sensor values as a function of cycles for four selected sensors (a)-(d) from four independent engine: original (blue) and reconstructed (red). The cycles are cumulative. The gray lines drawn to differentiate the units.
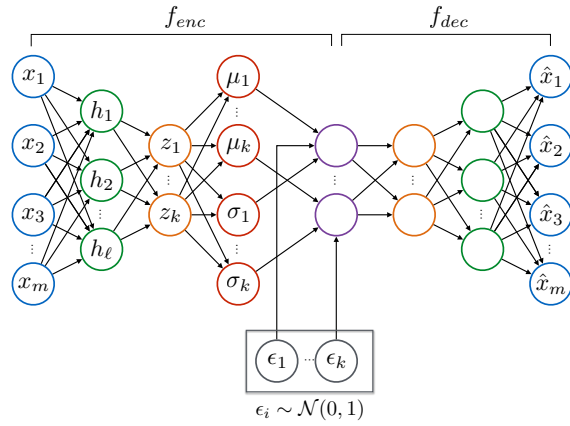

Figure 2: Overall architecture of VAE with an input dimension of m. $\epsilon \sim \mathcal{N}(0,1)$ indicates the reparameterization trick is used in the training of VAE.

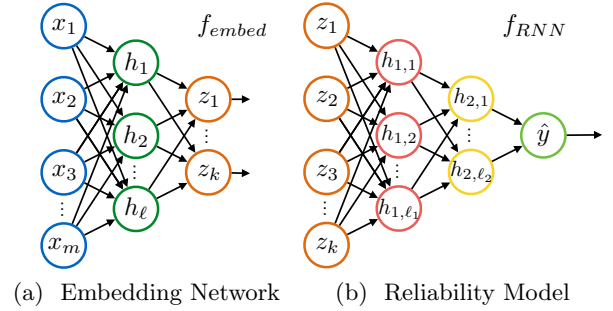(a) Embedding Network   (b) Reliability Model

Figure 3: Embedding followed by supervised learning in latent space. (a) embedding network is taken from the encoder of the trained VAE in Fig. 2. (b) reliability model is built upon the latent space defined by the embedding network. While RNN architecture is used, the recurrence relation is not shown for simplicity.

Section 2.2 and 2.3. We provide a description of the implemented architectures for RUL prediction and several important architectural decisions on design parameters in Section 4.

3.1.1 Non-linear Embedding

Embedding in semi-supervised learning is a procedure to map $\mathbf{x}$ in the original space into a lower dimensional space, $\mathbf{z}$, with the mapping obtained from unsupervised learning. The goal of embedding is to improve generalization of a supervised model. In our approach, we use the VAE to obtain the non-linear embedding. It is done by training the VAE as described in Section 2.3 using $\mathbf{x}$ from both labeled $\mathcal{D}_L$ and unlabeled data $\mathcal{D}_U$ to learn a mapping function f_{embed} for latent variable $\mathbf{z}$. The latent space is defined by the output layer chosen in the encoder of VAE as illustrated in Fig. 3(a). We choose k^{th} layer of the encoder with k less than $L_{encoder}$ (a total number of layers used in the encoder) to avoid using approximate samples from posterior distribution.

While for the parameterization of the approximate posterior in VAE, feed-forward neural network is conventionally chosen as originally suggested in [15]. In our case, we choose the RNN-architecture for the parameterization to model the temporal dependency of data. Incorporating the RNN architecture into VAE to include the latent random variables into the hidden state of a RNN requires the estimation of timestep-wise variational lower bound, which is not trivial. We instead simply introduce a weight factor α such that the two terms in Eq. 8 is differently weighted while modeling the temporal dependence in data with the RNN architecture.

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \alpha E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] \quad (9)$$

The weight factor is tuned considering the overall reconstruction efficiency of VAE.

3.1.2 Reliability Model

After obtaining the non-linear embedding from the VAE, we train a model (*reliability model*) with the RNN architecture in Eq. 4 using $(\mathbf{z}^{(i)}, y^i)$ where $\mathbf{z}^{(i)} = f_{embed}(\mathbf{x}^i)$ and $\mathbf{x}^i \in \mathcal{D}_L$. The resulting reliability model is used to predict RUL for a given set of historical CM data. It is noted the resulting model is similar to the *latent feature discriminative*
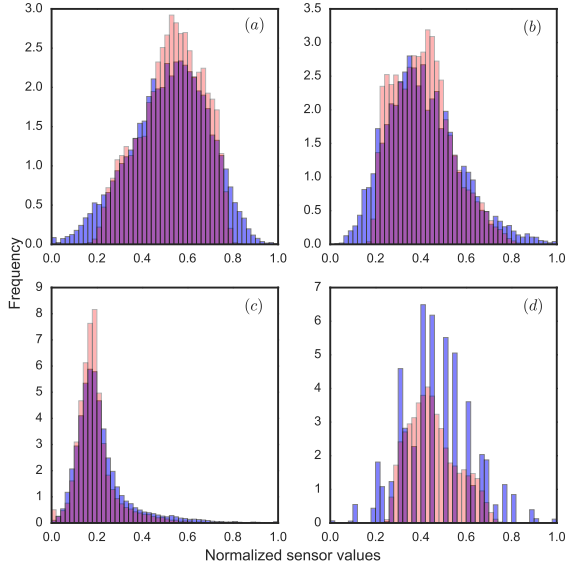
Figure 4: Normalized frequency of the values of selected sensors in Fig 1 (a)-(b) from all the engines in the dataset: original (blue) and reconstructed (red).

model (M1), which shows the state-of-the-art performance in various classification tasks [14]. The biggest difference lies in the way the non-linear embedding is obtained. The M1 model uses approximate samples from the posterior distribution as an embedding. In our case we rather use the kth layer output as an embedding as we want to avoid using the approximate sampling in order to achieve high precision RUL prediction[1]. The corresponding procedure is illustrated in the architectures of the embedding network and the reliability model in Fig. 3(a) and 3(b).

The entire procedure of our approach from generative modeling to supervised learning of reliability model is given in Algorithm 1, where the mini-batch $\mathbf{x}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$ is a randomly drawn sample of size M. The $\mathcal{L}$ indicates a corresponding loss function with $\widetilde{\mathcal{L}}$ implying that it is calculated for the data points in mini-batch as an approximation to the quantity for the entire data points. For the VAE, for example,

$$\mathcal{L}(\theta, \phi; \mathbf{x}) \simeq \widetilde{\mathcal{L}}(\theta, \phi; \mathbf{x}^M) = \frac{n}{M} \sum_{i=1}^{n} \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \qquad (10)$$

3.1.3 Self-learning Approach

Self-learning also known as self-labeling is an iterative algorithm that uses supervised learning in repetition [3]. As the name suggest, the algorithm tries to "bootstrap" using its own prediction. We use the simplest version of self-learning algorithm in order to see if it can help the generalization of supervised learning for the reliability prediction. It is done by first building a reliability model by training a RNN model using the labeled data, $(\mathbf{x}^{(i)}, y^i) \in \mathcal{D}_L$. The resulting model, so called *base learner*, is then used to make RUL estimations

---

[1]We tried with the approximate samples from the posterior distribution as features to train the reliability model and found the prediction precision is lowered because of the stochastic nature of sampling.

for the un-labeled data $\mathbf{x}^{(j)} \in \mathcal{D}_U$. The estimated RUL, $\widetilde{y}^{(j)}$, is then used in the next iteration of training such that the base learner is trained again with $(\mathbf{x}^{(i)}, y^i)$ and $(\mathbf{x}^{(j)}, \widetilde{y}^{(j)})$, treating $\widetilde{y}^{(j)}$ as real labels. The iteration is repeated until a certain convergence condition is met. In our study, we use a single iteration since we do not impose any convergence condition for the algorithm.

Algorithm 1: Semi-supervised learning based on non-linear embedding with VAE. We use settings M, $N = 100$ and $k = 3$. The first phase follows the minibatch version of the AEVB algorithm [15]

Phase 1 *Generative Modeling*

    Input: $\mathbf{x}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M \in \mathcal{D}$

1 $\theta, \phi \leftarrow$ Initialize parameters

2   **repeat**

3 $\mathbf{x}^M \leftarrow$ Random mini-batch of size, M

4 $\mathbf{z}^M \sim q_\phi(\mathbf{z}^M|\mathbf{x}^M)$

5 $\mathbf{g} \leftarrow \nabla_{\theta,\phi}\widetilde{\mathcal{L}}_{VAE}^M$

6 $\theta, \phi \leftarrow$ Update parameters using gradient $\mathbf{g}$ (e.g., Stochastic Gradient Descent)

7   **until** *convergence of parameters (θ,ϕ)*

8   **return** θ, ϕ

9 **end**

Phase 2 *Supervised Learning with Embedding*

    Input: $(\mathbf{x}^N, y^N) \in \mathcal{D}_L$ and θ

10 $\gamma \leftarrow$ Initialize parameters

11   Embedding $\mathbf{x}^N$ into the latent space defined by the kth output layer of the encoder:
     $\mathbf{z_k}^N \leftarrow f_{embed}(\mathbf{x}^N; \theta)$

12   **repeat**

13 $\mathbf{z}^N \leftarrow$ Random mini-batch of size, N

14 $y^N \sim f_{RNN}(\mathbf{x}^N; \gamma)$

15 $\mathbf{g} \leftarrow \nabla_\gamma \widetilde{\mathcal{L}}_{RNN}^N$

16 $\gamma \leftarrow$ Update parameters using gradient $\mathbf{g}$ (e.g., Stochastic Gradient Descent)

17   **until** *convergence of parameters (γ)*

18   **return** γ

19 **end**

---

4. EXPERIMENTAL RESULTS

This section presents the evaluation of our approach on a publicly available dataset, *Turbofan Engine Dataset*, provided by Prognostics Center of Excellence in National Aeronautics and Space Administration (NASA) [26]. The dataset is composed of simulated multi-sensor readings from a fleet of aircraft engines of the same type, created with a realistic damage propagation modeling based on C-MAPSS [27]. The dataset is further divided into training and testing sets. We use the training data to build models for RUL estimation and use the testing data to assess the performance of the models with different metrics. For the evaluations of the proposed approach, we consider the case where the true historical RUL is partially known in the training set. It is done by randomly dropping RUL information for a fraction of the engines in the training samples. We vary the fraction in order to cover a multitude of conditions.
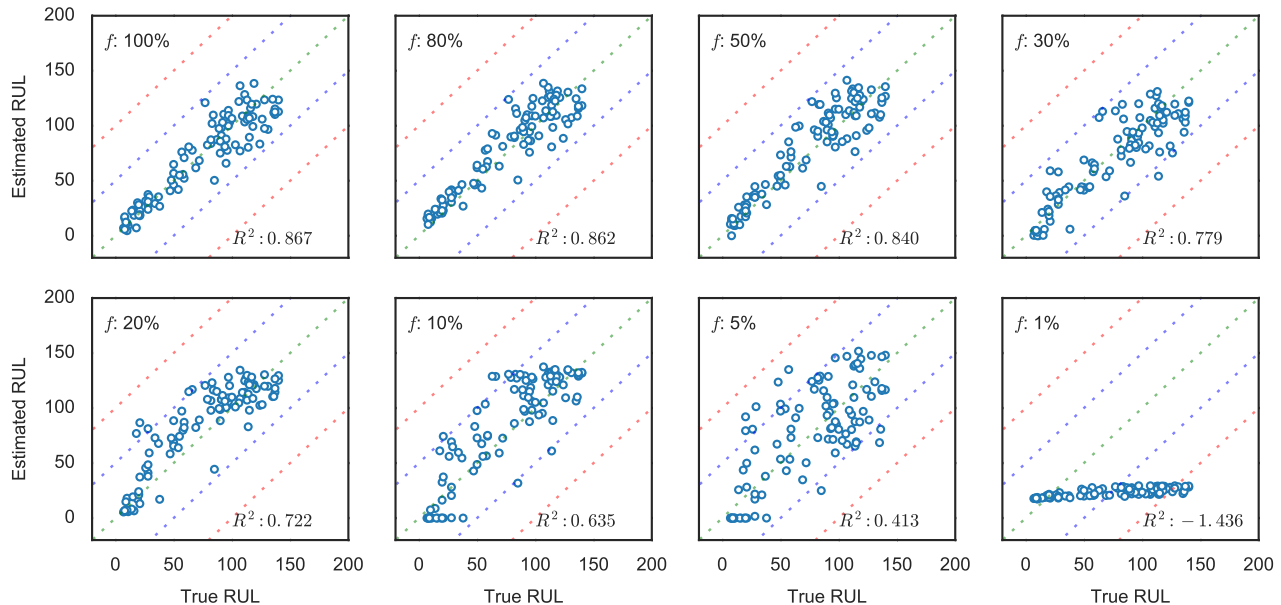
Figure 5: The actual and the estimated RULs in different labeled fraction scenarios. f is the labeled fraction. The green-dotted line indicates perfect estimation where the difference between the actual and the estimated RULs is zero. The blue and red dotted lines indicate the difference of 30 and 50 cycles, respectively. R^2 score is also shown.

4.1 C-MAPSS Turbofan Engine Dataset

The dataset, *FD001*, consists of multiple multivariate (24 sensors including 3 operational settings) time-series from a fleet of engines with different degree of initial wear and manufacturing variation that are unknown. The training and the testing sets contain 100 such engines for each. For the engines in the training set, run-to-failure time-series trajectories are provided. Whereas for the engines in the testing set, time-series trajectories are truncated prior to failures with true RUL values at the point of truncation given in a separate file for validation. A total number of cycles in the training set and in the testing set reach about 20K and 13K, respectively.

4.2 Evaluations

4.2.1 Overview

We directly model the engine failure by parameterizing the RNN architecture in Eq. 4 such that the input sensor readings are directly mapped to historical RUL. RUL is assigned to each sequence for a given engine by simply counting the number of cycles left before failure (by definition). In this way, we avoid to create any domain-specific assumptions for the modeling. We fist build a reliability model in a purely supervised setting, where all the training samples are used with their corresponding RULs. We then omit the RULs for a randomly chosen set of engines in the training set and repeat the procedure. The faction of engines, for which the RUL information is dropped, is varied from 100% down to 1% ($f = 100, 80, 50, 30, 20, 10, 5,$ and 1%). To remove any selection bias, we repeat the random selection five times for each fraction. The performance measures are then averaged over the five different sets of samples except for the 100% labeled scenario, where no selection is involved. Also in or-

der to take the model variability into account, we always use the homogeneous ensemble of the models obtained from five independent training, which in general produce a better result.

4.2.2 Preprocessing

Based on the 3 operational settings, we identify 14 different operational modes that are uniquely defined. We normalize the 21 sensor readings (excluding the 3 operational settings) before feeding into our models to train. The normalization is done for each sensor based on min-max scaling and it is done operational-mode-wise such that each sensor is scaled to have a range between 0 and 1 in each operational mode. The sensors that are constant (i.e., $\sigma_x = 0$) after the normalization are dropped. After checking carefully with its impact on the prediction accuracy, we further drop *discrete* sensors with the number of unique values falling below 20. For the sake of simplicity, the discrete variables were excluded in the training of VAE. For the RUL prediction, we limit the value below a maximum RUL of 140 cycles as suggested in [20, 7]. Consequently, we limit the assigned RULs in the training set by setting those that exceed the maximum RUL to be the maximum.

4.2.3 Metrics

Different performance metrics are employed to assess the performance of the models. This includes the mean squared error (MSE), mean absolute error (MAE), and the score metric proposed in [27]. The proposed score metric is defined such that late predictions are more heavily penalized than early prediction:

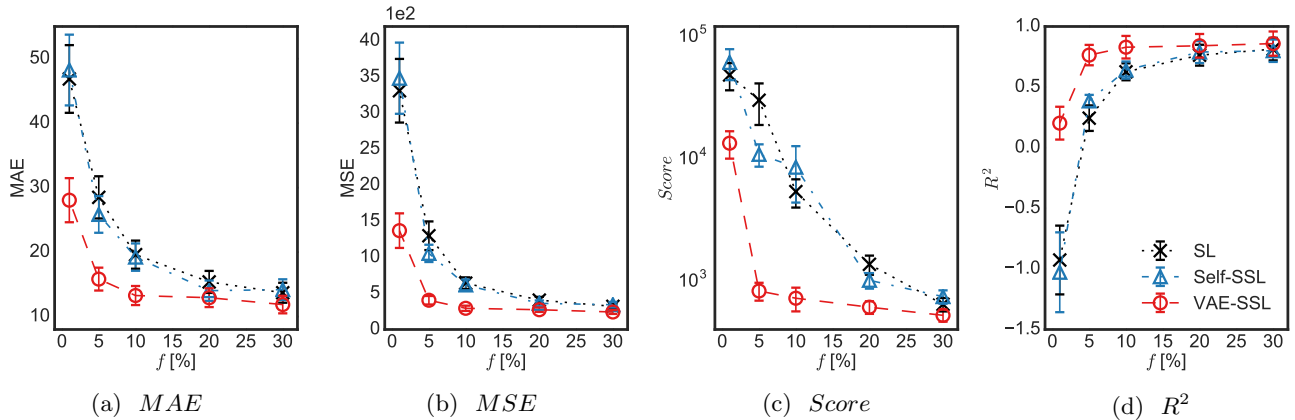$$S = \sum_{i=1}^{n} (\exp(\alpha|\Delta_i|) - 1) \tag{11}$$

Figure 6: Measured performance of different approaches as a function of the labeled fraction: (a) MAE, (b) MSE, (c) $Score$, and (d) R^2. The error bar indicates the standard error of mean at each point.

where Δ_i is the difference between the estimated and the true RUL values, i.e., estimated RUL - true RUL, for the ith engine in the testing set and α is equal to 1/13 if $\Delta_i < 0$ or 1/10 otherwise. We also use the coefficient of determination also known as R^2 score as a different measure of regression quality[2].

4.3 Results

4.3.1 Reconstruction with Variational Autoencoder

We use a three-layer architecture in the encoder (17-8-4) and in the decoder (4-8-17) with batch normalization applied to each layer except for the last layer. The architecture is chosen considering the reconstruction error and the performance of the non-linear embedding. The use of batch normalization is considered essential for training VAE with when the architecture becomes deep (i.e., $L > 2$) [11]. As mentioned earlier, we incorporate the RNN architecture into the VAE such that latent variables are modeled with the dynamic hidden state of a RNN architecture. This requires a modification in the objective function in Eq 8 so that sum of loss is averaged in temporal dimension. The model is trained end-to-end using the Adam optimizer [13] with a mini-batch size of 100.

Fig. 1 shows four sample sensor readings for several engines in sequence overlaid with the reconstructed sensor readings from VAE. In Fig. 4, we show the frequency of the values in histograms to see distributions. The results clearly demonstrate even with the one-fourth of the original input dimension in the bottleneck layer of VAE, the reconstruction is done reasonably well, capturing time dependent features of the signals. The sum of reconstruction loss and the KL-divergence stays below 0.3 when the trained VAE is used to reconstruct the validation sets[3]. This may support the manifold hypothesis that high dimensional data tend to lie in the vicinity of a low dimensional manifold [5].

4.3.2 Baseline Results

Fig. 5 shows the baseline results of RNN-based models with a four-layer architecture[4] trained in a purely supervised setting. It shows the true RUL in the x-axis and the estimated RUL in the y-axis in different labeled fraction scenarios. From the top left to the bottom right, the labeled fraction, f, is varied from 100% to 1%. When the full training set is used, the trained model shows good performance with the measured MSE, MAE, $Score$ and R^2 of 228, 11.3, 345, and 0.87, respectively. This result is comparable to that of other approaches [20, 1, 12, 23, 24].

Now as the fraction changes, the prediction accuracy decreases. It can clearly be seen from Fig. 5 that the correlation between the actual and the estimated RULs get worse and worse with decreasing fraction, also indicated by the R^2 score shown in the figure. The performance degradation is expected because the complexity of model requires a large amount of training data. In the current implementation of the RNN-based model, the number of tunable parameters reaches few hundreds. Nevertheless, the performance degradation is not prominent until the fraction is decreased down to 30%, indicating that only 30% of the training samples are already sufficient for the model to give a reasonably good result. As such, we rather focus in the region below 30%, where the performance degradation is significantly showing.

4.3.3 Semi-supervised Approach

As discussed in Section 3.1, we apply semi-supervised learning techniques to see if the performance can be improved when only a small fraction of the training data is labeled. We try two different semi-supervised learning techniques, i.e., self-learning and VAE-based non-linear embedding. Fig. 6 shows the results of the RUL estimation quantified by the three different metrics plotted as a function of f. From left to right, MAE, MSE, $Score$, and R^2, are plotted with corresponding statistical uncertainties. The score is plotted in logarithmic scale. It quantitatively shows the performance

---

[2]R^2 can yield negative values as models can be arbitrary worse.

[3]A fraction of the training data is kept from training and used for validation

[4]We adopt both GRU and LSTM architecture in the models. However, as we find no quantitative difference in performance, we choose the simpler architecture, namely, GRU for the final results.

Table 1: Comparison of different approaches. The best result is highlighted in bold.

Metric	Method	1%	5%	10%	20%	30%	100%
MAE	SL	46.9 ± 5.3	28.5 ± 3.3	19.7 ± 2.2	15.4 ± 1.7	13.7 ± 1.5	$\mathbf{11.3} \pm 1.3$
	Self-SSL	48.2 ± 5.5	25.8 ± 2.8	19.2 ± 2.1	14.0 ± 1.6	14.2 ± 1.6	$\mathbf{11.3} \pm 1.3$
	VAE-SSL	$\mathbf{28.1} \pm 3.4$	$\mathbf{15.8} \pm 1.8$	$\mathbf{13.3} \pm 1.5$	$\mathbf{12.9} \pm 1.5$	11.9 ± 1.4	$\mathbf{11.3} \pm 1.3$
MSE	SL	3309 ± 441	1299 ± 199	642 ± 80	407 ± 45	344 ± 35	228 ± 27
	Self-SSL	3483 ± 493	1055 ± 120	618 ± 74	362 ± 40	344 ± 38	228 ± 27
	VAE-SSL	$\mathbf{1370} \pm 240$	$\mathbf{405} \pm 59$	$\mathbf{294} \pm 36$	$\mathbf{274} \pm 31$	$\mathbf{243} \pm 27$	$\mathbf{221} \pm 26$
$Score\ [\times 10^{-2}]$	SL	482 ± 121	299 ± 111	53.6 ± 13.9	13.6 ± 2.43	6.42 ± 0.81	$\mathbf{3.45} \pm 0.41$
	Self-SSL	610 ± 173	109 ± 22.4	84.9 ± 41.3	10.0 ± 1.46	7.42 ± 0.93	$\mathbf{3.45} \pm 0.41$
	VAE-SSL	$\mathbf{133} \pm 34$	$\mathbf{8.26} \pm 1.38$	$\mathbf{7.20} \pm 1.58$	$\mathbf{6.09} \pm 0.72$	$\mathbf{5.22} \pm 0.59$	4.75 ± 0.56
R^2	SL	-0.93 ± 0.28	0.24 ± 0.11	0.63 ± 0.07	0.76 ± 0.09	0.81 ± 0.09	$\mathbf{0.87} \pm 0.10$
	Self-SSL	-1.03 ± 0.33	0.39 ± 0.05	0.64 ± 0.07	0.79 ± 0.09	0.80 ± 0.09	$\mathbf{0.87} \pm 0.10$
	VAE-SSL	$\mathbf{0.20} \pm 0.14$	$\mathbf{0.76} \pm 0.08$	$\mathbf{0.83} \pm 0.09$	$\mathbf{0.84} \pm 0.10$	$\mathbf{0.86} \pm 0.10$	$\mathbf{0.87} \pm 0.10$

degradation becomes increasingly larger as the fraction decreases. The result from the self-learning plotted in a triangular shape reveals that the performance is essentially the same within statistical uncertainty except for the score at f = 5%, where a distinctive improvement is shown.

Now, the result from the non-linear embedding plotted in a circle shows improvements for f below 30. The level of improvement becomes increasingly larger reaching x1.7, x2.4, and x3.6 improvement, respectively for MAE, MSE, and $Score$. The R^2 improvement is also notable. We note the prediction accuracy of the model is kept rather high even down to 5% fraction. The numerical compilations of the measured performance metrics are given in Table 1 for all three approaches: SL indicates supervised learning only. $Self\text{-}SSL$ is the self-learning. $VAE\text{-}SSL$ is the proposed method.

5. DISCUSSION

This paper described a semi-supervised learning approach to predict asset failures when there exist relatively small label information available. Our approach uses the non-linear embedding based on a deep generative model. We used the variational autoencoder as a deep generative model choice because of its relatively easy trainability on top of its firm theoretical foundation. The VAE was trained unsupervised while utilizing all the available data whether it is labeled or unlabeled. With this approach, we pursued achieving good prediction accuracy even when the available label information is highly limited.

The approach was extensively evaluated on a publicly available Turbofan Engine dataset by varying the fraction of labels down to 1%. The evaluation shows significant improvements in prediction accuracy when compared to the supervised and a naive semi-supervised approach, i.e., self-learning at 1%, the improvement reaches up to x3.6 for the proposed score metric. While the 1% fraction could be considered rather extreme, it is not unusual in practice since the acquisition of exact health status information that can be related to failures is exceedingly difficult in many cases. In image recognition tasks, order of 0.1 to 1% labeled fraction is typically considered for similar studies.

Overall, our study suggests the semi-supervised learning approach with the non-linear embedding based on deep generative model is effective when dealing with the problem of insufficient labels in future reliability prediction. Such problem is of tremendous practical interest in a wide range of data-driven PHM applications, especially when sophisticated deep learning architectures are used in modeling. In principle, there is little reason why our approach can not be extended to use other deep generative models for the embedding. In ensuing efforts, we are extending the approach to incorporate non-linear embedding using GAN.

6. REFERENCES

[1] G. S. Babu, P. Zhao, and X.-L. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International Conference on Database Systems for Advanced Applications*, pages 214–228. Springer, 2016.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Transaction of Neural Network*, 5(2):157–166, Mar. 1994.

[3] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[4] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, Oct. 2014.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[7] F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, pages 1–6. IEEE, 2008.

[8] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

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

[10] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*, 2015.

[11] C. Kaae Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. Ladder Variational Autoencoders. *ArXiv e-prints*, Feb. 2016.

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

(ISIE), 2014 IEEE 23rd International Symposium on, pages 2463–2468. IEEE, 2014.

[13] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, Dec. 2014.

[14] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014.

[15] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, Dec. 2013.

[16] Q. Z. Y. H. Kwok L. Tsui, Nan Chen and W. Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015(793161):17, 2015.

[17] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht. Iot-based prognostics and systems health management for industrial applications. *IEEE Access*, 4:3659–3670, 2016.

[18] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, London, UK, UK, 1998. Springer-Verlag.

[19] L. Maaløe, C. Kaae Sønderby, S. Kaae Sønderby, and O. Winther. Auxiliary Deep Generative Models. *ArXiv e-prints*, Feb. 2016.

[20] P. Malhotra, V. TV, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. *CoRR*, abs/1608.06154, 2016.

[21] R. K. Mobley. *An introduction to predictive maintenance*. Butterworth-Heinemann, 2002.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[23] Y. Peng, H. Wang, J. Wang, D. Liu, and X. Peng. A modified echo state network based remaining useful life estimation approach. In *Prognostics and Health Management (PHM), 2012 IEEE Conference on*, pages 1–7. IEEE, 2012.

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

[25] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672, 2015.

[26] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository*, 2008.

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

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

[29] J. Sikorska, M. Hodkiewicz, and L. Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, 25(5):1803–1836, 2011.

[30] P. Werbos. Backpropagation through time: what does it do and how to do it. In *Proceedings of IEEE*, volume 78, 1990.

[31] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao. Deep learning and its applications to machine health monitoring: A survey. *arXiv preprint arXiv:1612.07640*, 2016.