Dynamical Variational Autoencoders for Estimating the Remaining Useful Life of Machinery

Marco Star[1], and Kristoffer McKee[2]

[1] *Institute of Mechanical and Electrical Engineering, Centre for Industrial Mechanics, University of Southern Denmark, Sønderborg, 6400, Denmark*
mstar@sdu.dk

[2] *Faculty of Science and Engineering, School of Civil and Mechanical Engineering, Curtin University, Perth, 6102, Australia*
k.mckee@curtin.edu.au

ABSTRACT

The purpose of this work is to show the effectiveness of using generative models, specifically Dynamical Variational Autoencoders (DVAEs) for Remaining Useful Life (RUL) estimation. Many deep learning methods simply output point estimates of the RUL, generative models have the benefit of being optimized to learn an underlying probability distribution, this allows for uncertainty quantification. This work showcases how to construct a conditional DVAE to learn how to sample from $p(y_{1:T}|\mathbf{x}_{1:T})$, where $y_{1:T}$ is a sequence of RUL estimates and $\mathbf{x}_{1:T}$ are a sequence of sensor signals. It is shown why noncausal sensors are important when constructing this conditional model and how one can achieve state of the art results using DVAEs. Because the DVAE is a generative model and learns to sample from $p(y_{1:T}|\mathbf{x}_{1:T})$, one can also quantify the uncertainty of the RUL estimates directly with this model. This is tested on NASA's CMAPSS turbofan engine dataset and an open dataset from a dust filter experimental setup and it is demonstrated it is able to achieve state of the art results.

1. INTRODUCTION

Predictive maintenance techniques are used to help determine when maintenance should be performed based on the state of the machine. This helps one exploit the machine to its full potential while also preventing unexpected failures and downtime of equipment, giving economic and safety benefits to the stakeholders. An important component of predictive maintenance is machinery prognostics, often defined as the estimation of a machine's Remaining Useful Life (RUL). Hence, finding a model that can accurately estimate the RUL is important for the success of predictive maintenance as a whole. Another important consideration is the ability of this model to quantify the uncertainty. Uncertainty quantification is a vital aspect of RUL estimation, as there are always multiple sources of uncertainty that can affect the model and influence results; hence, estimating the uncertainty is an important component of a prognostics method (Sankararaman, 2015). Deep learning has shown a lot of promise in many different fields for solving a variety of problems that have an abundance of data, such as in image recognition, large language models and even in machinery prognostics using deep neural networks as models. While deep learning has shown the ability to achieve highly accurate RUL predictions, but it often outputs point estimates without quantifying the uncertainty (Peng et al., 2019) This work presents a model that shows it is capable of state-of-the-art performance while quantifying the uncertainty. It works using a generative model, which are not normally used for direct RUL estimation; hence, this work constructs a generative model for direct RUL estimation.

Currently, in the literature, various deep learning models are used to predict the RUL of machinery. For example, sensor signals can be transformed into a spectrogram image, and a Convolutional Neural Network (CNN) can be used to process these images and estimate the RUL (Zhu et al., 2019; X. Li et al., 2018). Recurrent Neural Networks (RNNs) are another popular network architecture that can be used to estimate the RUL from time-series sensor signals directly (Y. Zhang et al., 2018; Chen et al., 2021). Recently, some works have worked with attention mechanisms due to their success in other domains involving time-series (D. Xu et al., 2022; Q. Zhang, Yang, & Liu, 2024; Ragab et al., 2021; Q. Zhang, Liu, & Ye, 2024; K. Zhao et al., 2023). To include uncertainty quantification into these neural network models, a popular strategy is to use Monte Carlo (MC) Dropout (Peng et al., 2019; Cao et al., 2023; Lee & Mitici, 2023; Mitici et al., 2023; Xia et

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

https://doi.org/10.36001/IJPHM.2025.v16i2.4639

al., 2023). MC dropout is popular because it is easy to implement by adding dropout to the network. However, a downside is that while it accounts for epistemic (model/parameter) uncertainty, it does not account for aleatoric uncertainty (such as sensor noise) (Basora et al., 2025; Nemani et al., 2023). Another way to quantify uncertainty is to train the neural network to output the mean and variance estimates of the RUL. This effectively approximates a Gaussian distribution to represent the RUL (Kim & Liu, 2020; Z. Zhao et al., 2020). Similarly, parameters of other distributions like the log-normal distribution can be the outputs of the neural network (Nguyen et al., 2022). Another method models the inputs stochastically and then uses multiple samples as inputs to a neural network, which outputs the RUL to account for measurement uncertainty (Y. Gao et al., 2021). One can also estimate RUL uncertainty by developing a Health Indicator (HI) and extrapolating to a known threshold. If these HI trajectories are projected to the threshold using MC simulations, one can estimate a probability distribution of RULs (Xia et al., 2023). Probabilistic curve fitting can also be applied to the HI observations to assess RUL uncertainty (Guo et al., 2017). Unscented Kalman Filters have also been used to refine neural network predictions as new data is acquired (Chang et al., 2022). Gaussian processes can also be fit to the RUL estimate outputs of a neural network, which has been shown to work well and is effective at uncertainty quantification (Z. Xu et al., 2021; Nemani et al., 2023).

However, none of these explore using generative models to capture the underlying RUL probability distribution directly, i.e. using sensor signal inputs, directly output RUL samples. Generative models can use neural networks to learn how to sample from an underlying unknown probability distribution. Hence, they are a good candidate for not only estimating the RUL but finding the distribution $p(y_{1:T}|\mathbf{x_{1:T}})$ i.e. the probability distribution of the sequence of RULs ($y_{1:T}$) given the sensors ($\mathbf{x}_{1:T}$).

Generative models are explored for some prognostics applications. But many of these are for HI construction. The Variational Autoencoder (VAE) encodes the input variables into a lower-dimensional latent variable. This property can be leveraged to create a HI from multiple sensor signals (Ping et al., 2019; Qin et al., 2021; Wei et al., 2021; Yang et al., 2020; Remadna et al., 2022). Generative Adversarial Networks (GANs) can also be used by training a discriminator network to determine whether the sensor data comes from a healthy machine or not and use the output as a health indicator (Que et al., 2019). In a similar vein, generative models can be used to reconstruct healthy data; therefore, when attempting to reconstruct sensor data from a degrading machine, there is a large error. Some have used this reconstruction error as a HI (X. Li et al., 2020; Zhai et al., 2021; González-Muñiz et al., 2022).

In this work the focus is on using a generative model to directly estimate the RUL probability distribution given the sensor signals. To do this a class of models known as Dynamical Variational Autoencoders (DVAEs) are used (Girin et al., 2021). These classes of models have been used on other time-series tasks such as audio generation or modelling dynamical systems for control or reinforcement learning (Fraccaro et al., 2016, 2017; Hafner et al., 2019). Here we are less focused on the latent dynamics part of the model and more on constructing a model to estimate the conditional RUL distribution $p(y_{1:T}|\mathbf{x_{1:T}})$. However, as mentioned in Girin et al. (2021), there are many ways of structuring a DVAE, and so the question remains: how does one apply this class of model to the prognostics problem? This is what this work addresses.

In this work we aim to show how to construct a DVAE for RUL estimation given sensor signals related to degradation. It is shown that state-of-the-art results can be achieved if one structures the DVAE so that the conditional sensor signals are noncausal. Here this is achieved using a sequence-to-sequence model based on a sliding time window i.e. given a time window of sensor signals, the model returns a sequence of RUL outputs for that time window. This construction is shown to achieve state-of-the-art results on both the CMAPSS turbofan engine dataset Saxena & Goebel (2008) and a dust filter dataset from the Reliability Engineering & Prognostics and Health Management group at Esslingen University (Mauthe et al., 2021).

2. BACKGROUND

2.1. Variational Autoencoders

Variational Autoencoders (VAEs) are a generative deep learning method that aims to sample from the original data distribution $p(Y)$, where Y is the input data. However, $p(Y)$ is generally unknown. To get around this, generative models represent the probability distribution $p(Y, Z)$ where Z is a latent variable. This works by factoring the generative model, $p(Y, Z) = p(Y|Z)p(Z)$, hence, sampling the latent variable $Z^{(i)} \sim p(Z)$ then "decoding" using $Y \sim p(Y|Z^{(i)})$ would indirectly sample from the original data distribution $p(Y)$. This leads to the following integral,

$$p(Y) = \int p(Y, Z)dZ = \int p(Y|Z)p(Z)dZ. \quad (1)$$

If we want to use deep learning to train a VAE, we need to turn this into a loss function, which is often done by taking the negative logarithm of $p(Y)$ and minimising it,

$$-\log p(Y) = -\log \int p(Y|Z)p(Z)dZ. \quad (2)$$

However, the integral (Eq. 2) often has no analytical solution and cannot be solved directly. To solve this integral, the VAE uses importance sampling and defines an importance distri-

bution, $q(Z|Y)$. $q(Z|Y)$ is often called the inference model or the encoder, as it encodes the input variables (Y) into the latent space. The encoder is used to derive a loss function related to Eq. 2 known as the Evidence Lower BOund (ELBO). The ELBO indirectly optimises the original criterion and is stated as (Kingma & Welling, 2014),

$$\mathcal{L}(\theta_y, \theta_z, \phi, Y) = -\mathbb{E}_{q_\phi(Z|Y)}\left[\log p_{\theta_y}(Y|Z)\right] + D_{KL}\left(q_\phi(Z|Y)||p_{\theta_z}(Z)\right). \quad (3)$$

Where D_{KL} is the KL-divergence operator that provides a metric of how "close" the distributions $q_\phi(Z|Y)$ and $p_{\theta_z}(Z)$ are, and we have introduced the parameters ϕ, θ_y, and θ_z which represent the model parameters (weights and biases of the neural networks). Mathematically, the KL-divergence operator can be stated as,

$$D_{KL}\left(q(Z)||p(Z)\right) = \int q(Z)\log\frac{q(Z)}{p(Z)}dZ. \quad (4)$$

Note minimising $-\log\, p(Y)$ is the same as maximising the likelihood that the VAE samples Y from $p(Y)$, which is the goal of generative models. Hence, minimising the ELBO is an indirect way of solving the integral in Eq. 1 allowing the VAE to sample from $p(Y)$.

From the theory so far, it can be seen that there are three main components to the ELBO,

1. Decoder network: $p_{\theta_y}(Y|Z)$
2. Prior: $p_{\theta_z}(Z)$, for a VAE, this is usually defined as a standard normal distribution $\mathcal{N}(0, I)$
3. Encoder network: $q_\phi(Z|Y)$

These are optimized using the ELBO loss function (Eq. 3). After training one can generate new samples Y by sampling from the prior $Z^{(i)} \sim p_{\theta_z}(Z)$ and then using the decoder $Y^{(i)} \sim p_{\theta_y}(Y|Z^{(i)})$. Note that the encoder is not needed to generate the samples after training. The workings of the VAE training and generative model are shown more intuitively using the sketch shown in Figure 1.

2.2. Dynamical Variational Autoencoders

2.2.1. From VAE to DVAE

The VAE is generally used to generate static data Y, e.g. generating new images similar to the input images used to train the VAE. To generate sequential data that involve some underlying dynamics, Dynamical Variational Autoencoders (DVAEs) can be used instead to capture the underlying dynamic structure and is better suited to generate sequential data (Girin et al., 2021). DVAEs essentially replace the static input and latent variables (Y and Z) of the VAE with sequences, $Y = y_{1:T}$ and $Z = z_{1:T}$, where here they are denoted as sequences starting at time $t = 1$ and ending at some arbitrary time $t = T$. Hence, using the same structure as the VAE, the

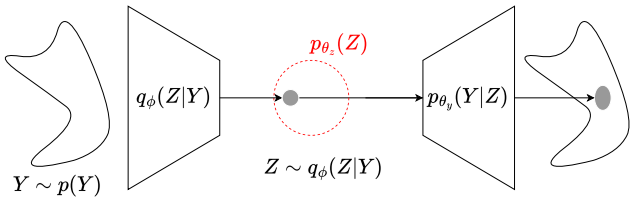


Figure 1. The VAE during training can be used to encode to a section of the prior distribution, during testing, the prior distribution can be sampled from (i.e. sample from the entire latent space) and this approximates a local Gaussian distribution in the input space $p(Y)$, which can be sampled from. Sampling from a larger area of $p(Y)$ can be done by sampling multiple latent variables via $Z \sim p(Z)$ and decoding them all to form local Gaussian approximations covering the distribution $p(Y)$.

generative and inference models would be,

$$\text{Generative model: } p_\theta(Y, Z) = p_\theta(y_{1:T}, z_{1:T}) \quad (5)$$

$$\text{Inference model: } q_\phi(Z|Y) = q_\phi(z_{1:T}|y_{1:T}). \quad (6)$$

This can be broken down further to describe the evolution of the variables along the time axis. Hence, Eqs. 5 and 6 can be broken down as,

$$\text{Generative model: } p_\theta(y_{1:T}, z_{1:T}) = \prod_{t=1}^{T} p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t})p_{\theta_z}(z_t|y_{1:t-1}, z_{1:t-1}) \quad (7)$$

$$\text{Inference model: } q_\phi(z_{1:T}|y_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1}, y_{1:T}). \quad (8)$$

Notice, from the generative model, the decoder and prior are found in the product (decoder: $p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t})$, prior: $p_{\theta_z}(z_t|y_{1:t-1}, z_{1:t-1})$). Analogous to the VAE, where the inference model is optimised to encode inputs Y to a latent space $p(Z)$, the DVAE inference model encodes the inputs to match some latent transition dynamics $p_{\theta_z}(z_t|y_{1:t-1}, z_{1:t-1})$. An important note is that the inference model is dependent on $y_{1:T}$ for all previous times, t, as the model cannot be factorised further to remove this dependency. Hence, even if the generative model is causal (current time values are only dependent on previous time values), the inference model is non-causal (future values are needed to estimate the current time values). Since the inference model is only used during training, this is not a problem as the entire training data sequence is available; therefore, the future values are available.

To train the DVAE, the new generative and inference models can be substituted into Eq 3, which results in the new loss

(Girin et al., 2021),

$$\mathcal{L}(\theta_y, \theta_z, \phi, y_{1:T}) = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T})}\left[\log p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t})\right] + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t-1}|y_{1:T})}[D_{KL}\left(q_\phi(z_t|z_{1:t-1}, y_{1:T})||p_{\theta_z}(z_t|y_{1:t-1}, z_{1:t-1})\right)]. \quad (9)$$

2.2.2. Conditional DVAE

In this paper, a conditional DVAE will be used. Hence, a sequence of conditional variables, $x_{1:T}$, will be introduced. Recall that generative models aim to sample from the original data distribution, i.e. $p(Y)$ for the VAE and $p(y_{1:T})$ for the DVAE; hence, a conditional DVAE is trained to sample from $p(y_{1:T}|x_{1:T})$. For machinery prognostics the conditional variables would be the sensor signals that indicate the state of the machine.

Adding, $x_{1:T}$, to the models in the DVAE gives,

$$\text{Generative model: } p_\theta(y_{1:T}, z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T}) \quad (10)$$

$$\text{Inference model: } q_\phi(z_{1:T}|y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T}) \quad (11)$$

Notice all the models are now noncausal if no further assumptions are made. Like with the VAE we can define a Decoder, Prior and Encoder. However, unlike the VAE the prior for the DVAE is modelled by a neural network.

$$\text{Decoder: } p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) \quad (12)$$

$$\text{Prior: } p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T}) \quad (13)$$

$$\text{Encoder: } q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T}) \quad (14)$$

The ELBO loss function becomes,

$$\mathcal{L} = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T}, x_{1:T})}\left[\log p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T})\right] + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t-1}|y_{1:T}, x_{1:T})}[D_{KL}\left(q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})||p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})\right)]. \quad (15)$$

This is derived in Appendix A.

3. METHODOLOGY

The goal of machinery prognostics is to estimate the RUL given some sensor signals from that machine or component. Mathematically, this can be stated as trying to find a model, $p(y_{1:T}|x_{1:T})$, i.e. finding the probability of RUL sequences, $y_{1:T}$, given sensor signals $x_{1:T}$. From Section 2.2.2, it can be seen the conditional DVAE is capable of sampling from a conditional distribution such as this one, i.e. $p(y_{1:T}|x_{1:T})$. Hence, to construct a probabilistic RUL estimation model, the input variables of the conditional DVAE can be set to the RUL sequences, $y_{1:T}$, and the conditional variables can be set to the sensor sequences $x_{1:T}$. The goal is to then train the networks from Eqs. 12, 13 and 14. In the following sections we discuss what network architectures are used, what assumptions are made and how a sequence such as $x_{1:T}$ is represented so it can be an input for a neural network. Note while the RUL is one dimensional, the latent and conditional variables are potentially multivariate so from now on they will be written using the bold notation to represent vectors, i.e. $\mathbf{z}_t$ and $\mathbf{x}_t$.

3.1. Constructing the Networks

One of the main problems here is the noncausal inputs, so at a current time $t \in [1, T]$ one must still find a way to represent $\mathbf{x}_{1:T}$ This is less of a problem for the Encoder as it is only used during the training stage, and the entire sequence of data is available during. However, both the Decoder and Prior are still used after training and they need to represent $\mathbf{x}_{1:T}$. A bidirectional RNN can be used to represent entire sequences as a single representative variable, $\overleftarrow{\mathbf{h}}_t := \mathbf{x}_{1:T}$ (Girin et al., 2021). Hence, to implement the DVAE, a sequence-to-sequence model will be used, where the bidirectional RNN will be used on a time window T of sensor data to represent $\mathbf{x}_{1:T}$, and this can be used in the Decoder and Prior for generating sequences $\mathbf{z}_{1:T}$ and $y_{1:T}$, where T now represents a hyperparameter selecting the size of a time-window.

When testing various assumptions for the DVAE it was found that removing the dependency on $y_{1:t-1}$ from the Prior and Decoder improved the performance. For ease of implementation, the Markov assumption is made for the latent variables in the generative model. The final models used can be stated as,

- Decoder: $p_{\theta_y}(y_t|\mathbf{z}_t, \mathbf{x}_{1:T})$
- Prior: $p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$
- Encoder: $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:T}, \mathbf{x}_{1:T})$

Both $y_{1:T}$ and $\mathbf{x}_{1:T}$ are represented using a bidirectional RNN for simplicity; specifically, a Gated Recurrent Unit (GRU) is used as the RNN cell. Future work could look at using other

network architectures to encode these sequences.

$$\overleftarrow{\mathbf{h}}_t^{(x)} = \text{BidirectionalRNN}(\mathbf{x}_{1:T}) \tag{16}$$

$$\overleftarrow{\mathbf{h}}_t^{(y)} = \text{BidirectionalRNN}(y_{1:T}) \tag{17}$$

The Encoder can also be modelled using an RNN (again, a GRU is used), and its hidden variables are inputs to a Multi-layered Perceptron (MLP) Network that is used to output the mean and log-variance of $\mathbf{z}_t$.

$$[\hat{\boldsymbol{\mu}}_{\mathbf{z}_t}, \log\hat{\boldsymbol{\sigma}}^2_{\mathbf{z}_t}] = \text{MLP}(\text{GRUCell}(\mathbf{h}_{t-1}, [\overleftarrow{\mathbf{h}}_t^{(y)}, \overleftarrow{\mathbf{h}}_t^{(x)}])), \tag{18}$$

where $\text{GRUCell}(\mathbf{h}_{t-1}, [\overleftarrow{\mathbf{h}}_t^{(y)}, \overleftarrow{\mathbf{h}}_t^{(x)}])$ outputs h_t using the previous hidden variable $\mathbf{h}_{t-1}$ that represents $\mathbf{z}_{1:t-1}$ and $[\overleftarrow{\mathbf{h}}_t^{(y)}, \overleftarrow{\mathbf{h}}_t^{(x)}]$ are the inputs to the GRUCell ($[\overleftarrow{\mathbf{h}}_t^{(y)}, \overleftarrow{\mathbf{h}}_t^{(x)}]$ denote a concatenation of the two variables). The outputs are $\hat{\boldsymbol{\mu}}_{\mathbf{z}_t}$ and $\log\hat{\boldsymbol{\sigma}}^2_{\mathbf{z}_t}$, are the mean and log-variance of a Gaussian distribution $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:T}, \mathbf{x}_{1:T})$. Both the Decoder and Prior are modelled using MLP networks.

$$[\boldsymbol{\mu}_{\mathbf{z}_t}, \log\boldsymbol{\sigma}^2_{\mathbf{z}_t}] = \text{MLP}\left([\mathbf{z}_{t-1}, \overleftarrow{\mathbf{h}}_t^{(x)}]\right), \tag{19}$$

$$[\boldsymbol{\mu}_{y_t}, \log\boldsymbol{\sigma}^2_{y_t}] = \text{MLP}\left([\mathbf{z}_t, \overleftarrow{\mathbf{h}}_t^{(y)}]\right). \tag{20}$$

Again, the inputs for the networks are concatenated so the network has a single input variable, and the outputs are the mean and log-variance of a Gaussian distribution representing the Prior and Decoder models. Hence, the models can now be stated as,

$$\text{Decoder: } p_{\theta_y}(y_t|\mathbf{z}_t, \mathbf{x}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{y_t}, \boldsymbol{\sigma}^2_{y_t}) \tag{21}$$

$$\text{Prior: } p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}_t}, \boldsymbol{\sigma}^2_{\mathbf{z}_t}) \tag{22}$$

$$\text{Encoder: } q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:T}, \mathbf{x}_{1:T}) = \mathcal{N}(\hat{\boldsymbol{\mu}}_{\mathbf{z}_t}, \hat{\boldsymbol{\sigma}}^2_{\mathbf{z}_t}) \tag{23}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ (diagonal matrix of variances in the multidimensional case). A graphical representation of the inference and generative models are shown in Figure 2.

3.2. Applying the Model

To train the conditional DVAE model, recall the ELBO from the background section (Eq. 15), which is stated here using the new model assumptions,

$$\begin{aligned}\mathcal{L} = &-\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|y_{1:T},\mathbf{x}_{1:T})}\left[\log p_{\theta_y}(y_t|\mathbf{z}_t, \mathbf{x}_{1:T})\right] \\ &+ \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|y_{1:T},\mathbf{x}_{1:T})}[\\ &D_{KL}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:T}, \mathbf{x}_{1:T})||p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})\right)].\end{aligned} \tag{24}$$

Notice that there are two main terms in this loss,

1. Negative Log-Likelihood (NLL): $-\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|y_{1:T},\mathbf{x}_{1:T})}\left[\log p_{\theta_y}(y_t|y_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{x}_{1:T})\right]$
2. KL-Divergence: $\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|y_{1:T},\mathbf{x}_{1:T})}[D_{KL}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:T}, \mathbf{x}_{1:T})||p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, y_{1:t-1}, \mathbf{x}_{1:T})\right)]$

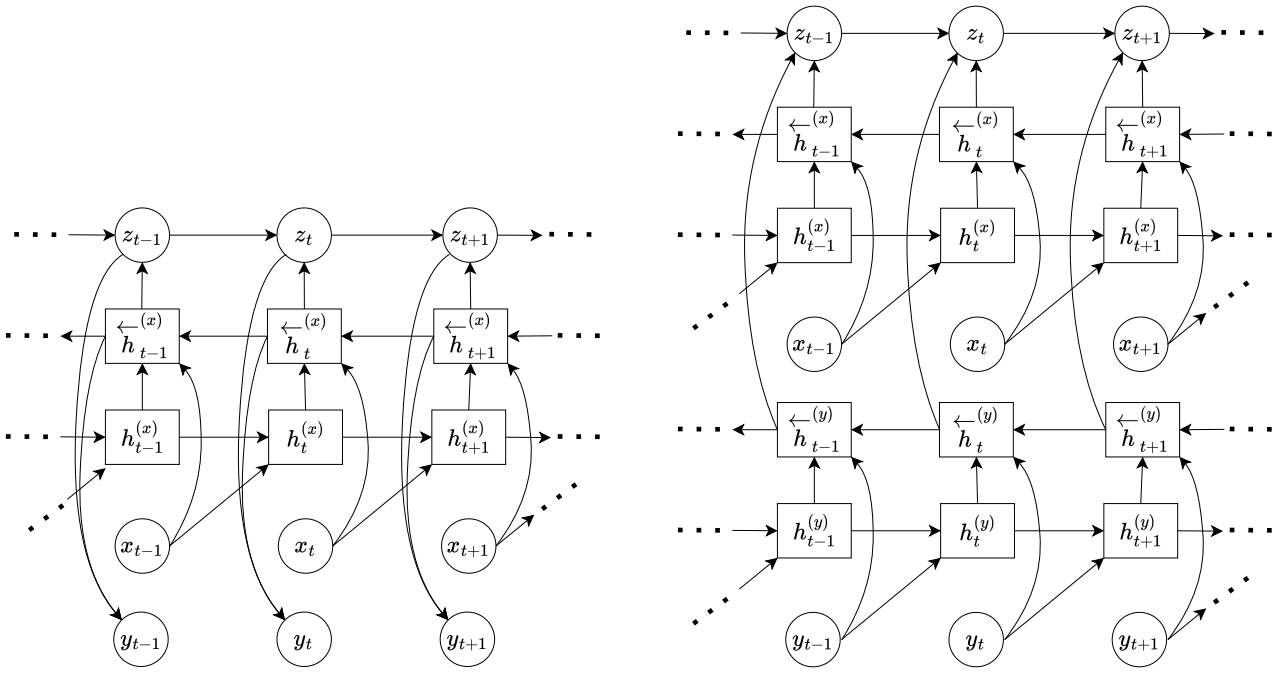
Hence, during training, the goal is to calculate these two terms and find the ELBO loss so that the deep learning software can optimise the model parameters based on the calculated loss. Figure 3 is a visual diagram that shows a high-level overview of the ELBO calculation when training the DVAE for RUL estimation. Note that after the ELBO is calculated, standard backpropagation-based training is used to optimise the network (Rumelhart et al., 1986).

The steps for applying the model after training are simpler as they involve applying the generative model and do not require the inference model,

1. Encode the sensor signal, $\mathbf{x}_{1:T}$, with the bidirectional RNN.
2. Initialise a latent variable $\mathbf{z}_0$ e.g. $\mathbf{z}_0 = 0$ (a tensor of zeros in this case)
3. Use the prior model to generate the next latent variable, $p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}_t}, \boldsymbol{\sigma}_{\mathbf{z}_t})$. i.e. starting from $\mathbf{z}_0$ generate the distribution for $\mathbf{z}_1$, sample $\mathbf{z}_1$ from that distribution and use that to generate $\mathbf{z}_2$ and so on.
4. From the last step, we end up with a sequence of sampled latent variables $\mathbf{z}_{1:T}$; use the samples at each time point in the decoder model, $p_{\theta_y}(y_t|\mathbf{z}_t, \mathbf{x}_{1:T}) = \mathcal{N}(\boldsymbol{\mu}_{y_t}, \boldsymbol{\sigma}_{y_t})$ to generate the distributions of the RUL estimates.
5. RUL estimates can sampled from the distribution, $y_t \sim p_{\theta_y}(y_t|\mathbf{z}_t, \mathbf{x}_{1:T})$
6. Monte Carlo simulation can be done through repeating steps 3-5 N times for some "large" value N (although all N calculations are done in parallel in this work).

The generative model is trained to sample from the distribution $p(y_{1:T}|\mathbf{x}_{1:T})$. Hence, by using Monte Carlo simulation and repeatedly applying the generative model and sampling N amount of $\mathbf{z}_{1:T}$ and $y_{1:T}$ sequences, the $y_{1:T}$ sequences represent samples that belong to $p(y_{1:T}|\mathbf{x}_{1:T})$ as the generative model has been trained to sample from this conditional distribution. With these samples $p(y_{1:T}|\mathbf{x}_{1:T})$ can be approximately represented and with larger N we can more accurately represent this distribution that captures what RUL sequences are most likely, given the observed sensor sequences.

Finally, to aid the training and help prevent overfitting, a β-DVAE is used. This is inspired by the β-VAE (Higgins et al., 2017), which multiplies the KL-divergence term of the ELBO loss by a constant β. The new altered loss is shown in Eq. 25.



(a) Generative model

(b) Inference model

Figure 2. A graphical representation of the conditional DVAE for RUL estimation, 2a is the generative model described by Eqs. 21 & 22 while 2b is the inference model described by Eq. 23. Note the variables inside circles are stochastic variables while the rectangles are deterministic

$$\mathcal{L} = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|y_{1:T},\mathbf{x}_{1:T})}\left[\log p_{\theta_y}(y_t|\mathbf{z}_t,\mathbf{x}_{1:T})\right] + \beta \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|y_{1:T},\mathbf{x}_{1:T})}[D_{KL}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1},y_{1:T},\mathbf{x}_{1:T})||p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{1:T})\right)]. \quad (25)$$

Hence, if β is above 1, the KL-divergence term has a larger weight than the negative log-likelihood term, which can help regularise the model. This was done as the model was prone to have accurate RUL estimates but struggled to quantify the uncertainty well.

To summarize, the DVAE approach is using a conditional-VAE that uses networks with a temporal component so variables are propagated through time. It is important for performance that the conditional variables are treated like non-causal variables as this will correspond to optimizing the marginal likelihood $p(y_{1:T}|\mathbf{x}_{1:T})$. Due to the VAE approach one can sample from the prior and use the decoder network to sample many $y_{1:T}$ estimates, thereby estimating the RUL and the certainty through the samples.

4. EXPERIMENTS

In this section, the data and experiment details are discussed. The code is also available on GitHub[1].

4.1. Data

4.1.1. CMAPSS

One of the datasets used for this experiment is NASA's Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dataset (Saxena & Goebel, 2008). CMAPSS is used as one of the datasets due to its popularity. It can, therefore, be easily compared to other similar methods to assess the performance of the DVAE. This dataset comprises simulated sensor readings from different turbofan engines that are run to failure for the training data and randomly stopped some time before failure for the testing data (with known RUL values so testing performance can be measured). There are four datasets included (each with training and testing data). Each dataset is meant to increase the difficulty of RUL estimation. Table 1 shows the different datasets and how they differ. The main differences that impact RUL estimation difficulty between the datasets are the number of operating conditions and fault/failure modes.

There are 26 total columns of data for each unit. The information contained in each column is shown in Table 2.

[1] https://github.com/StarMarco/DVAE_torch

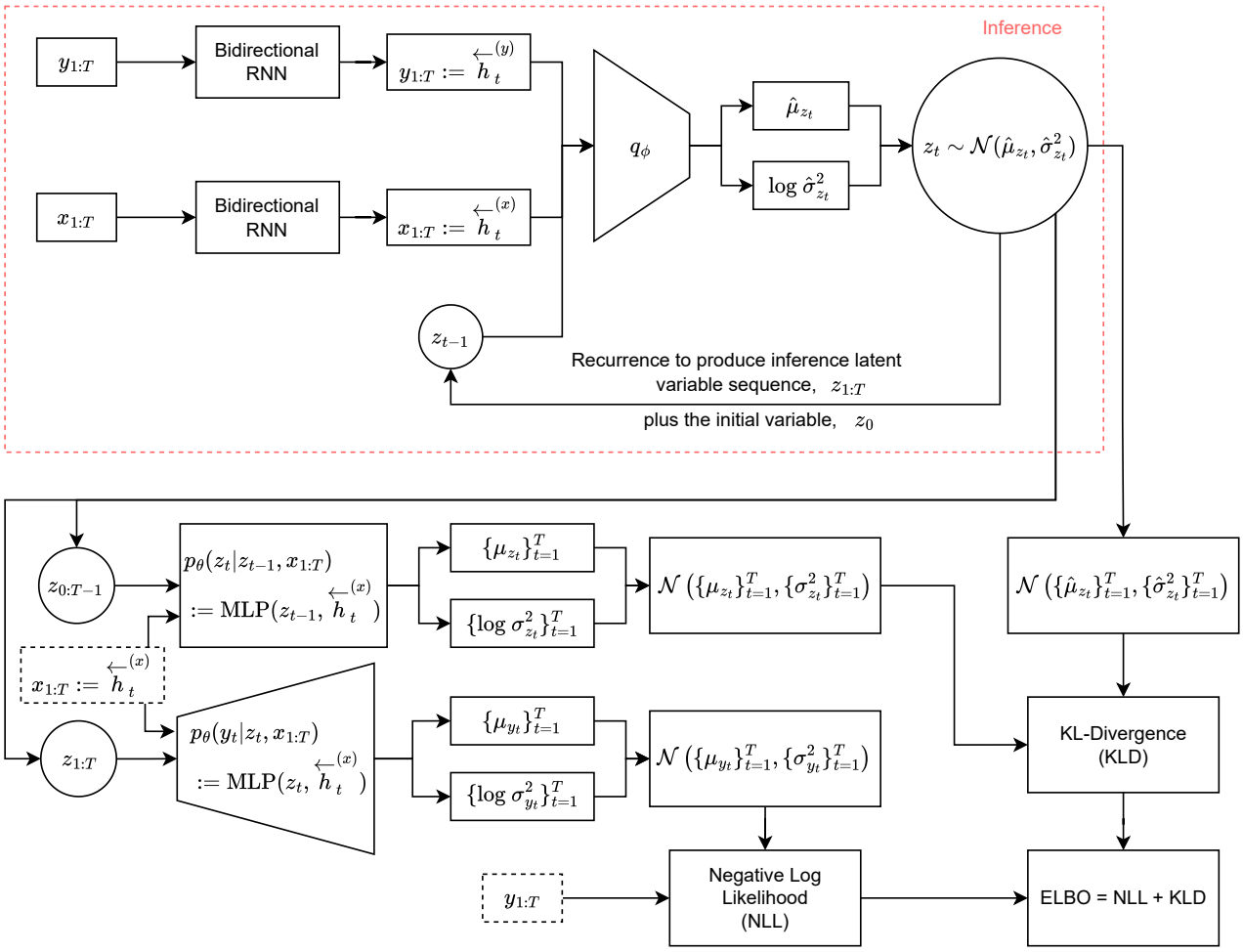


Figure 3. The block diagram for calculating the ELBO when training the DVAE for RUL estimation tasks. Shows how the data is processed during training in a simplified graphical manner. The inference block highlights a recursive step at time t, once this recursion goes through all times $t \in [1, T]$ we end up with the latent sequence $\mathbf{z}_{1:T}$. This can also include the initialised latent variable z_0 which can be initialised as a vector of zeros. Note the boxes made from dashed lines are simply copies of the data found in the inference block ($y_{1:T}$ and $\overleftarrow{\mathbf{h}}_t^{(x)}$), it is drawn again so arrows do not clutter the diagram.

Table 1. Different datasets found in CMAPSS: these state how many different engines are simulated, the number of operating conditions and the number of possible failure conditions they can experience (Fault Modes)

Dataset	Train Trajectories	Test Trajectories	Operating Conditions	Fault Modes
FD001	100	100	1	1
FD002	260	259	6	1
FD003	100	100	1	2
FD004	248	249	6	2

Table 2. Information contained in each column for the training and testing datasets

Columns	Data Contained
1	unit number
2	time (cycles)
3	operational setting 1
4	operational setting 2
5	operational setting 3
6-26	sensor measurements (1-21)

More detailed information on the CMAPSS dataset can be found in (Saxena et al., 2008).

4.1.2. Dust Filter Dataset with Varying Quality

The dust filter dataset published by the Reliability Engineering & Prognostics and Health Management group at Esslingen University is used as a second case study (Mauthe et al., 2021). The dataset is published as "Prognosis based on Varying Data Quality" on the website Kaggle[2]. It consists of data from multiple dust filters that are run to failure. Unlike the simulated CMAPSS dataset, this dataset was developed experimentally from a test rig. Hence, this helps show how the DVAE performs on data taken from a real-world setting and further validates the model. The training set contains the entire trajectories from start to end of life, while the test set is censored to stop at a random time before failure, much like CMAPSS. Sensors on the filters have a sampling rate of 10Hz and record the flow rate, particle feed and differential pressure across the filter. Failure was said to have occurred once the differential pressure first exceeded 600Pa. Hence, the main health indicator variable for the filter is the differential pressure, while the flow rate and the particle feed denote operating conditions. The documentation states that the two operating modes are based on 5/3-way and 3/2-way valves. All the data was gathered on the same filter test bench using the same sensors. The documentation also mentions that the measurement data was manipulated to simulate varying data quality that might arise due to different types of sensors being used or different sensor placement positions. Different signal-to-noise ratios were used for each differential pressure trajectory, and four different biases were introduced to simulate different sensor placements. Each differential pressure trajectory, therefore, has additional noise added, a different signal-to-noise ratio and is shifted by one of the four possible bias values. Hence, this dataset is suitable for testing a method like the DVAE, which seeks to quantify uncertainty.

[2]Link to the dataset, https://www.kaggle.com/prognosticshse/datasets

4.2. Preparing the Data

The sensor sequences for CMAPSS were used as the conditional input variables for the DVAE. To prepare these sensors for the DVAE model, firstly, the sensors that remain constant throughout the machine's lifetime were removed. Then the sequences were normalised based on what operating condition the sensors variables were in (Pasa et al., 2019). Eq. 26 shows how normalisation based on the operating condition is applied.

$$\hat{\mathbf{x}}_d^{(n)} = \sum_{c=0}^{N_c} \delta_c \odot \frac{\mathbf{x}_d^{(n)} - \mu_d^{(c)}}{\sigma_d^{(c)}}. \tag{26}$$

where $\mathbf{x}$ is the sensor signal, n is the unit number, d is the sensor number, $\hat{\mathbf{x}}$ is the normalised sensor signal, c is the operating condition, N_c is the total amount of operating conditions, $\delta_c = 1$ when the data is operating under operating condition c and $\delta_c = 0$ otherwise, and finally $\odot$ is element-wise multiplication. The operating conditions were identified using K-means clustering on the operational setting variables in the training dataset. Note that for FD001 and FD003, which only have a single operating condition, this equation reduces to standard normalisation that creates signals with zero mean and unit variance.

For the Dust Filter dataset, the input variables are the differential pressure, flow rate, dust feed, and the current time value. Here, the K-means clustering did not identify any meaningful operating conditions; hence, for simplicity, the sensors were normalised by scaling the values so the minimum value was 0 and the maximum value was 1. The formula for this is shown in Eq. 27.

$$\hat{\mathbf{x}}^{(n)} = \frac{\mathbf{x}^{(n)} - \min(\mathbf{x}^{(n)})}{\max(\mathbf{x}^{(n)}) - \min(\mathbf{x}^{(n)})}, \tag{27}$$

where $\max()$ and $\min()$ return the maximum and minimum values of an input sequence.

After these steps, both CMAPSS and the dust filter dataset are prepared in the same way to be compatible with the DVAE model. The model takes $\mathbf{x}_{1:T}$ and $y_{1:T}$ as noncausal inputs. To do this, a sequence-to-sequence approach was taken where the data was broken down into time-windowed portions using a sliding time window of size T. This sliding time window applied to the sensor data is illustrated in Figure 4. The different time-windowed slices of data were combined into batches when training the model; the size of the input tensors was (batch size, T, number of sensors). Hence, the model requires a time window of data points, $\mathbf{x}_{1:T}$, before it can estimate a time window of RUL estimates $y_{1:T}$. During the testing phase, the model is required to evaluate the entire se-

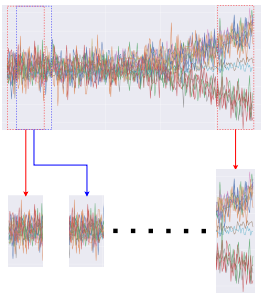


Figure 4. An illustration of a sliding time window applied to the sensor signals of a single unit in the CMAPSS dataset

quence of sensors for each testing unit, each of which may have a variable sequence length. To apply the sequence-to-sequence model in this setting, a sliding time window goes over the test units sensor sequence and uses the generative model on each time window to find a sequence of RUL estimates for each time window. These estimates are then put back together into a single sequence, and this is compared to the known target RUL values. Since the time windows have overlapping time points, the values with common time points are averaged when converting back into a sequence.

4.3. Hyperparameters

The hyperparameters that are chosen to train and evaluate the model can drastically affect the performance. Hence, it is important to optimise these hyperparameters to maximise the performance of the DVAE. Bayesian Optimisation (BO) was used for hyperparameter optimisation as it is more efficient than methods such as grid search. BO trains models on different input hyperparameter configurations and uses the validation loss as an output. Here the Python package Ax was used to perform BO and optimize the hyperparameters and it uses the package BoTorch as a backend (Balandat et al., 2020). The best hyperparameters are chosen based on the Negative Log-Likelihood and Mean Squared Error (MSE) metrics. The hyperparameters chosen from this process and the bounds of the search space for each hyperparameter are stated in Appendix B. Some baseline networks are trained for the Dust Filter dataset to compare their results with the DVAE. These networks also undergo the same hyperparameter optimisation process for fair comparison. The baseline network hyperparameters are also stated in the appendix. Section 5.2 gives more details about the baseline networks. Each network is trained for 200 epochs for both CMAPSS and Dust filter dataset experiments. Early stopping is used to choose the network configuration at the epoch with the best validation loss.

5. RESULTS

5.1. CMAPSS

This work compares the DVAE with other state-of-the-art RUL estimation methods for the CMAPSS dataset. Table 3 states the RMSE performance of the DVAE along with the other methods described previously. RMSE is defined as,

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}, \tag{28}$$

where N is the total amount of samples in the dataset over all the different time points in each testing unit, $\hat{y}_n$ is the RUL estimate and y_n is the corresponding true RUL value.

One of the main benefits of the DVAE is that it not only performs well in terms of RMSE accuracy but it can also quantify the RUL uncertainty. To assess this the α-coverage and α-mean metrics were used (Mitici et al., 2023). These can be stated as,

$$\alpha\text{-coverage} = \frac{1}{N}\sum_{n=1}^{N}\mathcal{I}(|\hat{y}_n^{0.5-0.5\alpha}, \hat{y}_n^{0.5+0.5\alpha}|), \tag{29}$$

$$\alpha\text{-mean} = \frac{1}{N}\sum_{n=1}^{N}\hat{y}_n^{0.5+0.5\alpha} - \hat{y}_n^{0.5-0.5\alpha}. \tag{30}$$

Where $\mathcal{I}(\cdot)$ is equal to 1 if the target RUL falls within the confidence interval defined by $|\hat{y}_n^{0.5-0.5\alpha}, \hat{y}_n^{0.5+0.5\alpha}|$ and is equal to 0 otherwise. $\hat{y}_n^{0.5-0.5\alpha}$ refers to the lower value of the RUL confidence interval (CI) and $\hat{y}_n^{0.5+0.5\alpha}$ refers to the higher value. Hence, if $\alpha = 0.95$ then $|\hat{y}_n^{0.5-0.5\alpha}, \hat{y}_n^{0.5+0.5\alpha}|$ defines the 95% CI and $\hat{y}_n^{0.5+0.5\alpha} - \hat{y}_n^{0.5-0.5\alpha}$ is the length of that CI. α-coverage is therefore an average of the points within the CI. This means we expect for any CI described by α, we should have the same proportion of RUL estimates within that CI. For example, if $\alpha = 0.9$, then α-coverage should ideally equate to 0.9, meaning 90% of the RUL estimates fall within the 90% CI. The α-mean gives an idea of how tight the CIs are. Note that α-coverage is equivalent to another metric sometimes used to evaluate uncertainty quantification methods known as the Predication Interval Coverage Percentage (PICP). However, the α term here makes it more clear what the upper and lower bounds of the CI are, and many works that use PICP only consider higher CIs such as 95% (Nguyen et al., 2022; G. Gao et al., 2020). These results are shown in Table 4 and compared to the Monte Carlo Dropout method used in (Mitici et al., 2023),

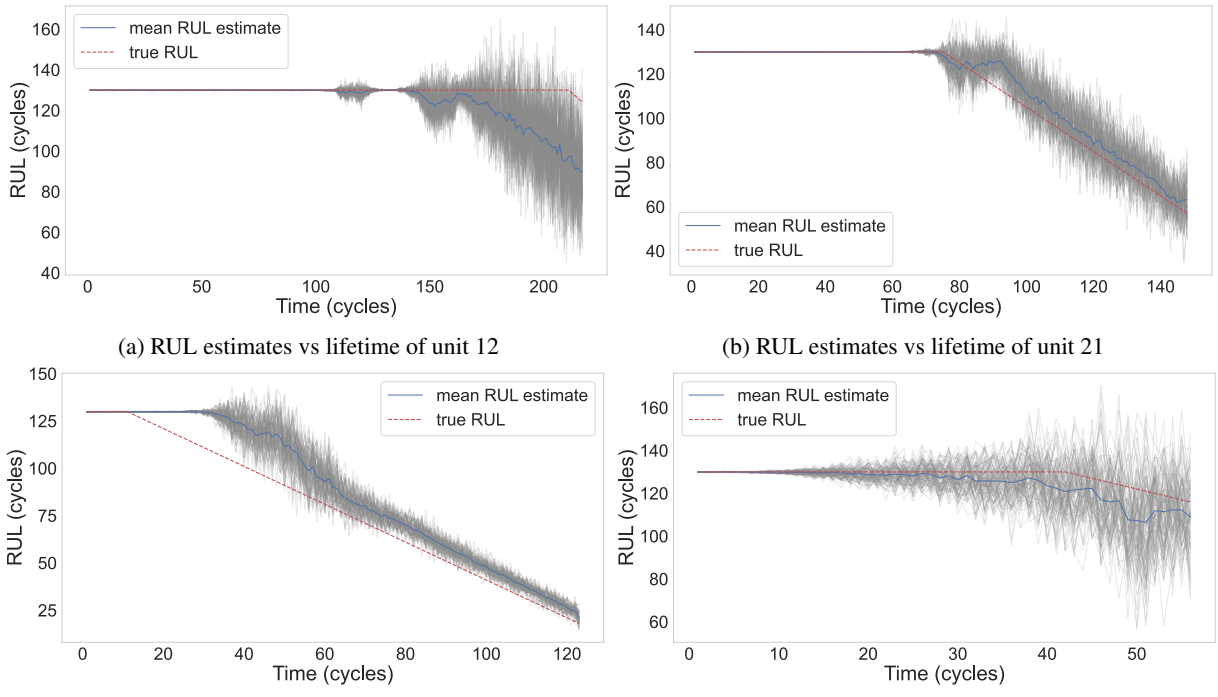
Figure 5 shows the plots of 100 sampled RUL trajectories, the mean RUL estimate, and the True RUL values over time. Units 12 and 41 of the FD001 dataset represent the units that stayed healthy for the longest and shortest amount of time respectively, while units 21 and 87 are randomly selected to illustrate the DVAE RUL estimation results.

Table 3. RMSE performance on each of the testing CMAPSS datasets

Method	FD001	FD002	FD003	FD004
DVAE (This work)	**6.89**	**8.23**	**6.99**	**7.78**
DL-NSGPR (Z. Xu et al., 2021)	7.4	11.8	7.5	8.3
CNN (X. Li et al., 2018)	12.61	22.36	12.64	23.31
TDDN (Qin et al., 2022)	9.47	10.93	9.17	11.16
DS-SANN (D. Xu et al., 2022)	11.64	13.34	12.28	14.98
SUR-TSMAE (Fu et al., 2022)	14.46	21.1	17.16	22.61
MCTAN (Ren et al., 2023)	11.69	16.41	10.72	17.36
DCCL (X. Li et al., 2022)	7.4	11.6	7.1	11.6
ST-GS4D (Wu et al., 2025)	11.18	14.37	11.38	14.04
DS-STFN (Q. Zhang, Yang, & Liu, 2024)	10.92	13.77	10.1	15.53
TCAT (Jiangyan et al., 2024)	11.12	13.40	11.02	17.56
CNN MC dropout with RL (Lee & Mitici, 2023)	11.81	14.49	11.69	17.73
CNN MC dropout (Mitici et al., 2023)	12.42	13.72	12.16	15.95

Table 4. The α-coverage and α-means for $\alpha = 0.95, 0.9, 0.5$, i.e. $95\%, 90\%$ and 50% confidence intervals, using a maximum RUL of 130. Note the closer the α-coverage is to its corresponding α value the better while the α-mean gives an idea of the average CI size

α	Metric	FD001		FD002		FD003		FD004	
		Ours	Dropout	Ours	Dropout	Ours	Dropout	Ours	Dropout
0.95	α-coverage	0.94	0.95	0.92	0.89	0.94	0.97	0.92	0.9
	α-mean	17.5	46.4	16.9	42.4	14.2	47.6	13.2	47.3
0.9	α-coverage	0.91	0.91	0.89	0.85	0.92	0.92	0.89	0.85
	α-mean	14.9	39.2	14.3	36.1	12.0	40.3	11.2	40.1
0.5	α-coverage	0.48	0.54	0.64	0.51	0.63	0.57	0.64	0.52
	α-mean	6.20	16.3	5.97	15.2	4.98	16.7	4.66	16.8



(a) RUL estimates vs lifetime of unit 12

(b) RUL estimates vs lifetime of unit 21

(c) RUL estimates vs lifetime of unit 41

(d) RUL estimates vs lifetime of unit 87

Figure 5. The RUL estimates over the known lifetime of the testing units for a selected subset of units in the FD001 test dataset. 100 trajectories are sampled to show the possible trajectories the DVAE can generate and give an idea of the distribution of the output RUL region and if the true RUL falls within this region.

To further illustrate how well the model captures the uncertainty, the final time RUL estimates for each of the units in the FD001 test dataset are plotted along with their 95% confidence intervals. This can give an indication of how well the model captured the uncertainty. It can also be seen that the testing units with high final RUL targets (the machines are still relatively healthy) have higher uncertainty bounds compared to the units closer to failure which have smaller bounds. The plot confirms what one might intuitively expect; as the machine gets closer to failure, the uncertainty is smaller, and the model grows more certain a failure will occur soon.

5.2. Dust Filter Dataset

A similar analysis of the DVAE is done for the Dust Filter dataset. However, to the author's knowledge, there are currently not many published works that have used this dataset. Therefore, in this section, the DVAE's performance will be compared to other popular networks to see how well the DVAE performs in comparison. This experiment is also a good sanity check to see how well the DVAE performs on another dataset and one that is not simulated like CMAPSS. The baseline networks used here are the bidirectional GRU, bidirectional Long Short Term Memory (LSTM) network and a standard feedforward MLP network. The bidirectional networks are useful comparisons as the DVAE uses bidirectional RNNs to encode the sensor signals. Here, the DVAE used a bidirectional LSTM encoder as the baseline LSTM performed the best on this dataset. Hence, it would make a good comparison to see how the bidirectional LSTM's performance changes when incorporated into the DVAE structure. Table 5 shows the RMSE results on the dust filter dataset.

Table 5. RMSE performance on the Dust Filter Dataset

Method	Dust Filter RMSE
DVAE (This work)	8.99
GRU	10.1
LSTM	9.14
MLP	11.0

The α-coverage and α-means for $\alpha = 0.95, 0.9, 0.5$ for the DVAE on the Dust Filter dataset are shown in Table 6 below.

Table 6. The α-coverage and α-means for $\alpha = 0.95, 0.9, 0.5$, i.e. $95\%, 90\%$ and 50% confidence intervals. Note the closer the α-coverage is to its corresponding α value the better and the α-mean gives an idea of the average CI size.

α	Metric	DVAE score
0.95	α-coverage	0.92
	α-mean	33.8
0.9	α-coverage	0.88
	α-mean	28.7
0.5	α-coverage	0.53
	α-mean	11.9

Some sample plots of the RUL estimates over time are also shown in Figure 7. The plot with the highest and lowest final RUL value in the test set are plotted along with two randomly chosen dust filters in the test set. Unit 20 has the highest final RUL (38.2) and is the farthest from failure while unit 19 has the shortest (17.1) and is closest to failure.

The final RUL estimate 95% confidence intervals are also plotted in Figure 8. For this test set, the final RULs are all similar, so we expect less of a range of confidence interval lengths compared to CMAPSS shown in Figure 6. In the CMAPSS case, there were some units far from failure and some very close to failure; hence, the units far from failure typically have large confidence intervals, while the ones closer to failure become more certain, as reflected by the smaller confidence interval.

6. DISCUSSION

Looking at the RMSE performance in Tables 3 and 5, the DVAE achieved state-of-the-art performance for CMAPSS and outperformed the baseline models for the Dust Filter dataset. Another method that performed well on CMAPSS while accounting for uncertainty was DL-NSGPR. The advantage of the DL-NSGPR method is that it only uses a relatively simple MLP network trained on the data for RUL estimation, and it uses Non-Stationary Gaussian Process Regression (NSGPR) on the MLP output RUL estimates to account for uncertainty. However, DVAEs are also not restricted to Gaussian distributions to represent the RUL estimates like DL-NSGPR is. The DVAEs generative model can be used to sample multiple $y_{1:T}$ estimates and represent any arbitrary probability distribution using those samples. Since Gaussian distributions are not always a valid assumption in RUL estimation tasks (Sankararaman, 2015), this property of DVAEs is beneficial.

Another strength of this DVAE model is that it provides a clear underlying theoretical framework. Looking at the theory behind DVAEs, others can use the same theory while finding different ways to achieve the underlying mathematical structure of the model. For example, instead of using a bidirectional RNN to encode $\mathbf{x}_{1:T}$ into a hidden variable, another network architecture could be used, such as a Transformer (Vaswani et al., 2017). Or perhaps use different assumptions than the ones used here, e.g. use transition model $p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ instead of $p_{\theta_z}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$. A useful aspect of using these sequential conditional generative modelling techniques, such as DVAEs, is that there can be some flexibility in how each model is constructed (e.g. what network architectures are used) while still guided by the overarching framework. One can see from the model breakdowns in Eqs. 10 and 11 that the DVAE makes it clear how important it is to have noncausal input variables. The utilization of this knowledge allowed for the effective sampling of the underlying distribution, $p(y_{1:T}|\mathbf{x}_{1:T})$, and the prediction

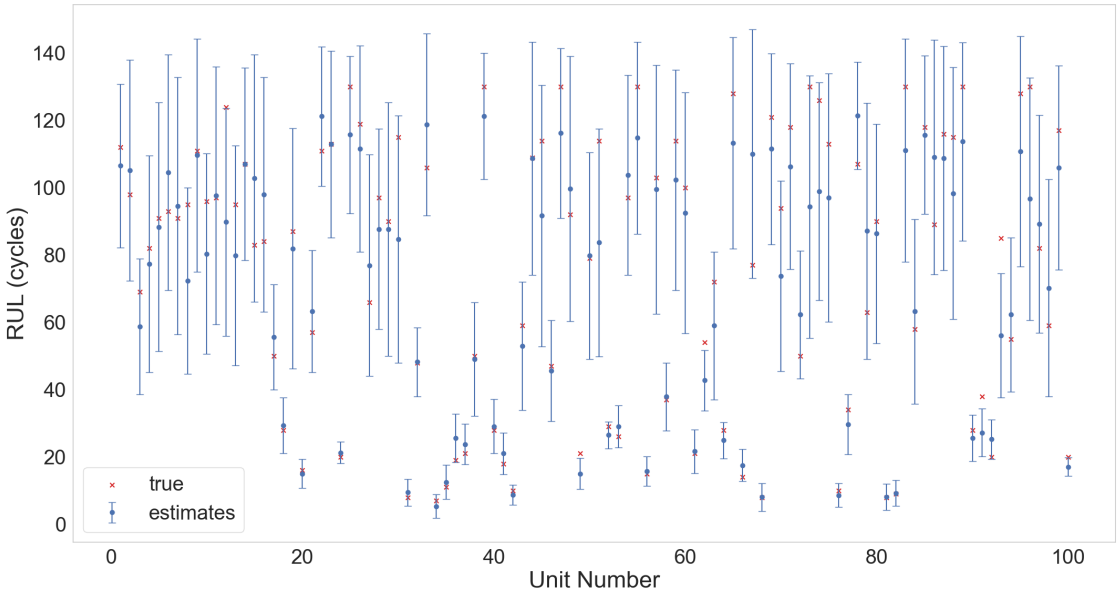
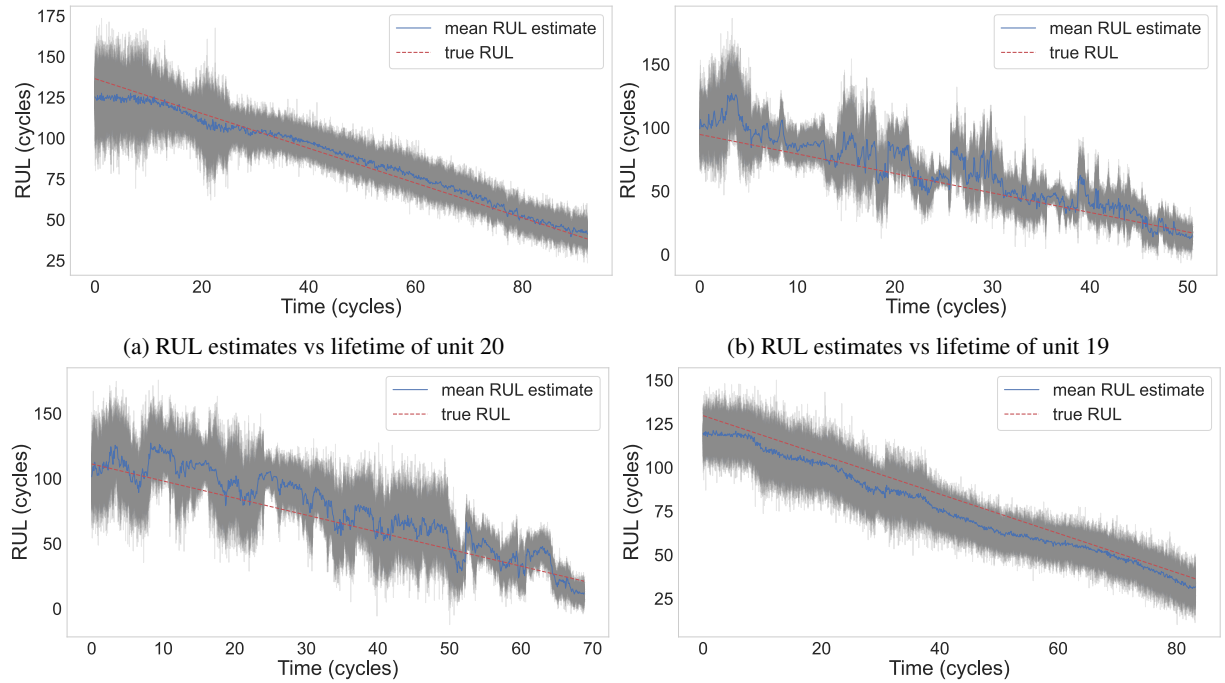


Figure 6. The final RUL mean estimates and True RUL values plotted as a scatter plot vs the unit index for each of the units in the FD001 test dataset. The 95% confidence intervals are also shown for the final RUL estimate.



(a) RUL estimates vs lifetime of unit 20

(b) RUL estimates vs lifetime of unit 19

(c) RUL estimates vs lifetime of unit 3

(d) RUL estimates vs lifetime of unit 8

Figure 7. The RUL estimates over the known lifetime of the testing units for a selected subset of units in the dust filter test dataset.

of the RUL. The experiments showed that this allowed the DVAE to achieve excellent results on the RMSE for the RUL predictions compared to the ground truth values. The actual network architectures stated here are less important and were

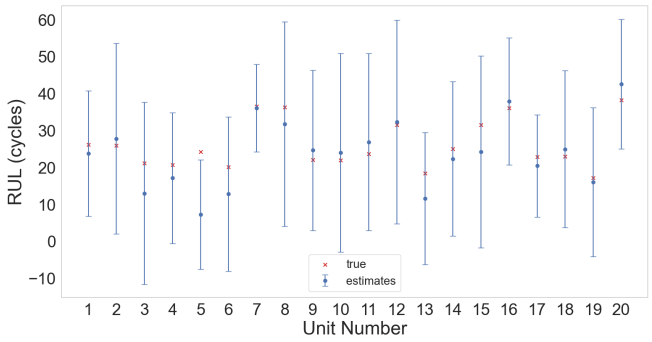


Figure 8. The final RUL mean estimates and True RUL values plotted as a scatter plot vs the index representing each dust filter in the test set. The 95% confidence intervals are also shown for the final RUL estimate.

mainly chosen to test the model's effectiveness using relatively simple network architectures used in other DVAE models (Girin et al., 2021). Hence, the adoption of the DVAE here shows an alternate way of quantifying the uncertainty and utilising existing network architectures for RUL estimation.

Figures 5a and 5c illustrated some edge cases in the FD001 test dataset. Unit 12 stays in the healthy region the longest, but the model's mean RUL estimates start the degradation process earlier than the true RUL shows. This may be due to the assumption that the machine starts to degrade around $RUL = 130$. While on average, this assumption may be true, for the edge cases such as units 12 and 41 where the machine stays healthier or starts degrading sooner than the other units, this assumption may fall apart. The max RUL of 130 essentially assumes degradation starts at 130 cycles for each unit, with constant RUL values representing a healthy machine. This may not be the case for all engines; however, more sophisticated methods of first detecting the degradation stage and then utilizing the RUL prediction algorithm is outside the scope of this work. Other methods also utilize this maximum RUL method and so to keep analysis fair this is utilized here as well. Hence, while the maximum RUL assumption may be wrong for some units and cause these inaccuracies as shown in Figures 5c and 5a, the maximum RUL is still kept for fair comparison to other methods in the literature.

Tables 4 and 6 indicate the DVAE's performance regarding uncertainty quantification. For the CMAPSS dataset, while the network performed well for $\alpha = 0.95$ and $\alpha = 0.9$, it overestimated the values for $\alpha = 0.5$ (except for the FD001 dataset). A possible explanation for the poorer performance on that metric may be that the marginal likelihood is not directly used but estimated through the ELBO criterion. This means the model is not directly being optimised using the marginal likelihood, $p(y_{1:T}|\mathbf{x}_{1:T})$. Hence, the final model does not accurately represent the marginal distribution. For example, A. H. Li et al. (2021) used an Extended Kalman filter to find a more direct estimate of the marginal likelihood, resulting in smoother trajectories. A potential way to test this is to use a particle filter framework to work directly with the marginal likelihood as DVAEs essentially use the particle filtering framework already (Naesseth et al., 2018; Maddison et al., 2017). However, this has its own issues, such as the particle filter requiring a resampling step, which is typically not differentiable. Hence, we cannot train the deep learning model using backpropagation if a step is not differentiable. Some works attempt to address this through a differential resampling step, but they also have their trade-offs (Corenflos et al., 2021; Jonschkowski et al., 2018; Lai et al., 2021). To keep this initial work simple, we used the ELBO formulation of the model, but this particle filtering approach could be explored in future works to test this hypothesis.

Finally, it is worth comparing the DVAE to some popular uncertainty methods, such as MC dropout. MC dropout only models the uncertainty of the model itself, otherwise known as epistemic uncertainty (Kim & Liu, 2020; Z. Zhao et al., 2020). The other type of uncertainty to consider is the aleatoric uncertainty, which can be defined as the inherent uncertainty of the experiment itself. In this setting, the aleatoric uncertainty would be the uncertainty of the measurements/sensors and operating conditions. The DVAE model learns how to sample from the distribution $p(y_{1:T}|\mathbf{x}_{1:T})$ by breaking the problem down into latent/state dynamics and a measurement/observation model. As mentioned in the previous paragraph, this is mathematically equivalent to a particle filtering setup. Note, in a particle filter, the latent dynamics and observation model are often modelled as distributions or with additive noise representing the model and measurement uncertainty respectively. Hence, it can be seen that by following this framework, one ends up modelling epistemic uncertainty through the uncertainty in state dynamics (latent state dynamics in our case) and aleatoric uncertainty through the observation model. However, in this work the sensors where not the observations, instead they were the conditional variables that were deterministically encoded using an RNN. Better accounting for the sensor noise could be another avenue of future work that could be done to improve this model.

7. CONCLUSIONS

This paper has showcased the effectiveness of deep generative models in machinery prognostics for RUL estimation. By using these noncausal models in the DVAE framework, we were able to train a model that could sample from the underlying data distribution $p(y_{1:T}|\mathbf{x}_{1:T})$ and come up with probabilistic RUL estimates. The DVAE achieved state-of-the-art performance on CMAPSS and performed better than the baseline models on the Dust Filter dataset. On the Dust Filter dataset and FD001 dataset in CMAPSS, the model was able to quantify the uncertainty well, as shown by the α-coverage results. However, it struggled for some of the $\alpha = 0.5$ on the other

CMAPSS datasets. We have outlined some potential reasons as to why this could be the case, such as the indirect optimisation of the marginal likelihood or not explicitly modelling the uncertainty of the sensor signals. However, this work shows that the DVAE could be a fruitful avenue to explore due to these preliminary results, which show they are capable of state-of-the-art performance.

A useful property of the DVAE is that through the repeated sampling of $y_{1:T}$, the distribution $p(y_{1:T}|\mathbf{x}_{1:T})$ can be expressed without needing to assume a specific probability distribution. For example, DL-NSGPR uses Gaussian Process Regression to quantify the uncertainty of the RUL estimates; however, this would restrict the distribution to a Gaussian distribution. Another interesting aspect about DVAEs is that from 7, it can be seen the DVAE can be interpreted as a Sequential Importance Sampling approach. Hence, future work could look at how the performance and uncertainty estimates could be improved if this approach is used when training the model.

The work presented here could potentially be improved or expanded upon by focusing on some topics such as,

- Different encoders for sequences (such as $\mathbf{x}_{1:T}$): For example, Attention Mechanisms could be used instead of the RNN or bidirectional RNN.
- Marginal Log-Likelihood loss via. Particle Filtering: Particle Filtering algorithms could potentially be adapted to directly estimate the marginal likelihood during training instead of indirectly optimising the network parameters using ELBO. The main problem with this is most Particle Filters require a resampling stage which is not differentiable (so it cannot be optimised using backpropagation in the deep learning software). Still, some work has been done to remedy this, which can be explored (Corenflos et al., 2021; Lai et al., 2021).
- Different models: The Prior and Decoder were simple MLP networks in this work. Other network architectures could be used to try to improve the performance of the DVAE.
- Improved latent dynamics: One could focus on trying to improve the latent dynamics of this method. This could potentially be used to improve the interpretability of the model through the use of the latent dynamics as a HI.
- Semi-supervised methods: The DVAE could be trained on the sensor values in an unsupervised manner. This could help pre-train networks for a supervised DVAE like the one used in this paper or help account for the uncertainties in the sensor values. This would be beneficial for reducing the amount of run-to-failure data needed to train an effective deep learning model.

REFERENCES

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). Botorch: A framework for efficient monte-carlo bayesian optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 21524–21538). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf

Basora, L., Viens, A., Chao, M. A., & Olive, X. (2025, 1). A benchmark on uncertainty quantification for deep learning prognostics. *Reliability Engineering & System Safety*, *253*, 110513. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S0951832024005854 doi: 10.1016/j.ress.2024.110513

Cao, L., Zhang, H., Meng, Z., & Wang, X. (2023). A parallel gru with dual-stage attention mechanism model integrating uncertainty quantification for probabilistic rul prediction of wind turbine bearings. *Reliability Engineering & System Safety*, *235*, 109197. Retrieved from https://www.sciencedirect.com/science/article/pii/S0951832023001126 doi: https://doi.org/10.1016/j.ress.2023.109197

Chang, Y., Zou, J., Fan, S., Peng, C., & Fang, H. (2022, 9). Remaining useful life prediction of degraded system with the capability of uncertainty management. *Mechanical Systems and Signal Processing*, *177*, 109166. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S0888327022003235 doi: 10.1016/j.ymssp.2022.109166

Chen, Z., Wu, M., Zhao, R., Guretno, F., Yan, R., & Li, X. (2021). Machine remaining useful life prediction via an attention-based deep learning approach. *IEEE Transactions on Industrial Electronics*, *68*, 2521-2531. doi: 10.1109/TIE.2020.2972443

Corenflos, A., Thornton, J., Deligiannidis, G., & Doucet, A. (2021, 7). Differentiable particle filtering via entropy-regularized optimal transport. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (Vol. 139, pp. 2100–2111). PMLR. Retrieved from https://proceedings.mlr.press/v139/corenflos21a.html

Fraccaro, M., Kamronn, S., Paquet, U., & Winther, O. (2017). A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in Neural Information Processing Systems 30, NIPS*.

Fraccaro, M., Sø nderby, S. r. K., Paquet, U., & Winther, O. (2016). Sequential neural models with stochastic layers. In D. Lee, M. Sugiyama, U. Luxburg,

I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29, p. 2207–2215). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2016/file/208e43f0e45c4c78cafadb83d2888cb6-Paper.pdf

Fu, S., Zhong, S., Lin, L., & Zhao, M. (2022, 12). A novel time-series memory auto-encoder with sequentially updated reconstructions for remaining useful life prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *33*, 7114-7125. doi: 10.1109/TNNLS.2021.3084249

Gao, G., Que, Z., & Xu, Z. (2020). Predicting remaining useful life with uncertainty using recurrent neural process. *Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS-C 2020*, 291-296. doi: 10.1109/QRS-C51114.2020.00057

Gao, Y., Wen, Y., & Wu, J. (2021, 1). A neural network-based joint prognostic model for data fusion and remaining useful life prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 117-127. doi: 10.1109/TNNLS.2020.2977132

Girin, L., Leglaive, S., Bie, X., Diard, J., Hueber, T., & Alameda-Pineda, X. (2021, 8). Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends® in Machine Learning*, *15*(1-2), 1–175. Retrieved from https://arxiv.org/abs/2008.12595http://www.nowpublishers.com/article/Details/MAL-089 doi: 10.1561/2200000089

González-Muñiz, A., Díaz, I., Cuadrado, A. A., & García-Pérez, D. (2022, 8). Health indicator for machine condition monitoring built in the latent space of a deep autoencoder. *Reliability Engineering & System Safety*, *224*, 108482. doi: 10.1016/J.RESS.2022.108482

Guo, L., Li, N., Jia, F., Lei, Y., & Lin, J. (2017). A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, *240*, 98-109. Retrieved from http://dx.doi.org/10.1016/j.neucom.2017.02.045 doi: 10.1016/j.neucom.2017.02.045

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019, 6). Learning latent dynamics for planning from pixels. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 2555–2565). PMLR. Retrieved from https://proceedings.mlr.press/v97/hafner19a.html

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., ... Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *Iclr* (p. 1-22).

Jiangyan, Z., Ma, J., & Wu, J. (2024). A regularized constrained two-stream convolution augmented Transformer for aircraft engine remaining useful life prediction. *Engineering Applications of Artificial Intelligence*, *133*, 108161. Retrieved from https://doi.org/10.1016/j.engappai.2024.108161 doi: 10.1016/j.engappai.2024.108161

Jonschkowski, R., Rastogi, D., & Brock, O. (2018, June). Differentiable particle filters: End-to-end learning with algorithmic priors. In *Proceedings of robotics: Science and systems* (pp. 1–9). Pittsburgh, Pennsylvania. doi: 10.15607/RSS.2018.XIV.001

Kim, M., & Liu, K. (2020). A bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions*, *53*, 326-340. Retrieved from https://www.tandfonline.com/doi/abs/10.1080/24725854.2020.1766729 doi: 10.1080/24725854.2020.1766729

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *CoRR*, *abs/1312.6114*.

Lai, J., Domke, J., & Sheldon, D. (2021). Variational marginal particle filters. *Proceedings of The International Conference on Artificial Intelligence and Statistics (AISTATS)*. Retrieved from https://par.nsf.gov/biblio/10359646

Lee, J., & Mitici, M. (2023). Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics. *Reliability Engineering & System Safety*, *230*, 108908. Retrieved from https://www.sciencedirect.com/science/article/pii/S0951832022005233 doi: https://doi.org/10.1016/j.ress.2022.108908

Li, A. H., Wu, P., & Kennedy, M. (2021). Replay overshooting: Learning stochastic latent dynamics with the extended kalman filter. In *2021 ieee international conference on robotics and automation (icra)* (p. 852-858). doi: 10.1109/ICRA48506.2021.9560811

Li, X., Ding, Q., & Sun, J. Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*(December 2017), 1–11. Retrieved from https://doi.org/10.1016/j.ress.2017.11.021 doi: 10.1016/j.ress.2017.11.021

Li, X., Wong, T.-K. L., Chen, R. T. Q., & Duvenaud, D. (2020, 1). Scalable Gradients for Stochastic Differential Equations. *arxiv*. Retrieved from http://arxiv.org/abs/2001.01328

Li, X., Zhang, W., Ma, H., Luo, Z., & Li, X. (2022, 10). Degradation alignment in remaining useful life prediction using deep cycle-consistent learning. *IEEE Transactions*

on Neural Networks and Learning Systems, *33*, 5480-5491. doi: 10.1109/TNNLS.2021.3070840

Maddison, C. J., Lawson, J., Tucker, G., Heess, N., Norouzi, M., Mnih, A., ... Teh, Y. (2017). Filtering variational objectives. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30, p. 1-11). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/fa84632d742f2729dc32ce8cb5d49733-Paper.pdf

Mauthe, F., Hagmeyer, S., & Zeiler, P. (2021). Creation of publicly available data sets for prognostics and diagnostics addressing data scenarios relevant to industrial applications. *International Journal of Prognostics and Health Management*. Retrieved from https://api.semanticscholar.org/CorpusID:244036445

Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic rul prognostics: The case of turbofan engines. *Reliability Engineering & System Safety*, *234*, 109199. Retrieved from https://www.sciencedirect.com/science/article/pii/S095183202300114X doi: https://doi.org/10.1016/j.ress.2023.109199

Naesseth, C., Linderman, S., Ranganath, R., & Blei, D. (2018, 4). Variational sequential monte carlo. In A. Storkey & F. Perez-Cruz (Eds.), *Proceedings of the twenty-first international conference on artificial intelligence and statistics* (Vol. 84, pp. 968–977). PMLR. Retrieved from https://proceedings.mlr.press/v84/naesseth18a.html

Nemani, V., Biggio, L., Huan, X., Hu, Z., Fink, O., Tran, A., ... Hu, C. (2023). Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. *Mechanical Systems and Signal Processing*, *205*, 888-3270. Retrieved from https://doi.org/10.1016/j.ymssp.2023.110796 doi: 10.1016/j.ymssp.2023.110796

Nguyen, K. T., Medjaher, K., & Gogu, C. (2022, 6). Probabilistic deep learning methodology for uncertainty quantification of remaining useful lifetime of multi-component systems. *Reliability Engineering and System Safety*, *222*. doi: 10.1016/j.ress.2022.108383

Pasa, G., Medeiros, I., & Yoneyama, T. (2019). Operating Condition-Invariant Neural Network-based Prognostics Methods applied on Turbofan Aircraft Engines. In *Annual conference of the phm society* (Vol. 11, pp. 1–10). doi: https://doi.org/10.36001/phmconf.2019.v11i1.786

Peng, W., Ye, Z.-S., & Chen, N. (2019). Bayesian Deep Learning based Health Prognostics Towards Prognostics Uncertainty. *IEEE Transactions on Industrial Electronics*, 1–1. Retrieved from https://ieeexplore.ieee.org/document/8681720/ doi: 10.1109/TIE.2019.2907440

Ping, G., Chen, J., Pan, T., & Pan, J. (2019). Degradation feature extraction using multi-source monitoring data via logarithmic normal distribution based variational autoencoder. *Computers in Industry*, *109*, 72–82. doi: 10.1016/j.compind.2019.04.013

Qin, Y., Cai, N., Gao, C., Zhang, Y., Cheng, Y., & Chen, X. (2022, 2). Remaining useful life prediction using temporal deep degradation network for complex machinery with attention-based feature extraction. *ArXiv*. Retrieved from http://arxiv.org/abs/2202.10916 doi: 10.48550/arxiv.2202.10916

Qin, Y., Zhou, J., & Chen, D. (2021). Unsupervised health Indicator construction by a novel degradation-trend-constrained variational autoencoder and its applications. *IEEE/ASME Transactions on Mechatronics*. doi: 10.1109/TMECH.2021.3098737

Que, Z. J., Xiong, Y., & Xu, Z. G. (2019, 12). A semi-supervised approach for steam turbine health prognostics based on gan and pf. *IEEE International Conference on Industrial Engineering and Engineering Management*, 1476-1480. doi: 10.1109/IEEM44572.2019.8978717

Ragab, M., Chen, Z., Wu, M., Kwoh, C.-K., Yan, R., & Li, X. (2021, 11). Attention-based sequence to sequence model for machine remaining useful life prediction. *Neurocomputing*, *466*, 58-68. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S0925231221013801 doi: 10.1016/j.neucom.2021.09.022

Remadna, I., Terrissa, L. S., Masry, Z. A., & Zerhouni, N. (2022). Rul prediction using a fusion of attention-based convolutional variational autoencoder and ensemble learning classifier. *IEEE Transactions on Reliability*. doi: 10.1109/TR.2022.3190639

Ren, L., Liu, Y., Huang, D., Member, S., Huang, K., & Yang, C. (2023). Mctan: A novel multichannel temporal attention-based network for industrial health indicator prediction. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, *34*. Retrieved from https://doi.org/10.1109/TNNLS.2021.3136768. (Uses attention mechanisms to weight differnet sensors in each channel based on how much the contribute to degradation. The attention also act temporally to deal with long sequence lengths.) doi: 10.1109/TNNLS.2021.3136768

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536. Retrieved from https://www.nature.com/articles/323533a0 doi: https://doi.org/10.1038/323533a0

Sankararaman, S. (2015). Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction. *Mechanical Systems and Signal Processing*, *52-53*, 228-247. Retrieved from https://www.sciencedirect.com/science/article/pii/S0888327014002052 doi: https://doi.org/10.1016/j.ymssp.2014.05.029

Saxena, A., & Goebel, K. (2008). *Turbofan Engine Degradation Simulation Data Set.* NASA Ames Research Center. Retrieved from https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008, 10). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (Vol. 41, pp. 1–9). IEEE. Retrieved from http://www.embase.com/search/results?subaction=viewrecord{&}from=export{&}id=L71115746{%}5Cnhttp://dx.doi.org/10.1515/jpm-2013-2003{%}5Cnhttp://sfx.aub.aau.dk/sfxaub?sid=EMBASE{&}issn=03005577{&}id=doi:10.1515{%}2Fjpm-2013-2003{&}atitle=Growth+factors+in+pregnancies+comp doi: 10.1109/PHM.2008.4711414

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Wei, Y., Wu, D., & Terpenny, J. (2021, 12). Learning the health index of complex systems using dynamic conditional variational autoencoders. *Reliability Engineering & System Safety*, *216*, 108004. doi: 10.1016/J.RESS.2021.108004

Wu, X., Liu, Z., & Wang, L. (2025, 4). Spatio-temporal degradation model with graph neural network and structured state space model for remaining useful life prediction. *Reliability Engineering and System Safety*, *256*. doi: 10.1016/j.ress.2024.110770

Xia, Y., Yao, W., Zheng, X., & Gong, Z. (2023, 3). Reliability analysis of heat source layout temperature field prediction considering uncertainty in deep neural network surrogate models. *Quality and Reliability Engineering International*. Retrieved from https://onlinelibrary.wiley.com/doi/full/10.1002/qre.3313https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.3313https://onlinelibrary.wiley.com/doi/10.1002/qre.3313 doi: 10.1002/QRE.3313

Xu, D., Qiu, H., Gao, L., Yang, Z., & Wang, D. (2022, 6). A novel dual-stream self-attention neural network for remaining useful life estimation of mechanical systems. *Reliability Engineering & System Safety*, *222*, 108444. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S0951832022001090 doi: 10.1016/j.ress.2022.108444

Xu, Z., Guo, Y., & Saleh, J. (2021). Accurate Remaining Useful Life Prediction With Uncertainty Quantification: A Deep Learning and Nonstationary Gaussian Process Approach. *IEEE Transactions on Reliability*. doi: 10.1109/TR.2021.3124944

Yang, T., Wang, J., Chen, L., Cui, Z., Qi, J., & Wang, R. (2020). Machinery health prognostics of dust removal fan data through deep neural networks. In H. Ning & F. Shi (Eds.), *Cyberspace data and intelligence, and cyber-living, syndrome, and health* (pp. 27–37). Singapore: Springer Singapore. doi: https://doi.org/10.1007/978-981-33-4336-8_3

Zhai, S., Gehring, B., & Reinhart, G. (2021, 3). Enabling predictive maintenance integrated production scheduling by operation-specific health prognostics with generative deep learning. *Journal of Manufacturing Systems*. doi: 10.1016/J.JMSY.2021.02.006

Zhang, Q., Liu, Q., & Ye, Q. (2024). An attention-based temporal convolutional network method for predicting remaining useful life of aero-engine. *Engineering Applications of Artificial Intelligence*, *127*, 952-1976. Retrieved from https://doi.org/10.1016/j.engappai.2023.107241 doi: 10.1016/j.engappai.2023.107241

Zhang, Q., Yang, P., & Liu, Q. (2024, 10). A dual-stream spatio-temporal fusion network with multi-sensor signals for remaining useful life prediction. *Journal of Manufacturing Systems*, *76*, 43-58. doi: 10.1016/j.jmsy.2024.07.004

Zhang, Y., Xiong, R., He, H., & Pecht, M. G. (2018). Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, *67*(7), 5695–5705. doi: 10.1109/TVT.2018.2805189

Zhao, K., Jia, Z., Jia, F., & Shao, H. (2023). Multi-scale integrated deep self-attention network for predicting remaining useful life of aero-engine. *Engineering Applications of Artificial Intelligence*, *120*, 105860. Retrieved from https://doi.org/10.1016/j.engappai.2023.105860 doi: 10.1016/j.engappai.2023.105860

Zhao, Z., Wu, J., Wong, D., Sun, C., & Yan, R. (2020). Probabilistic remaining useful life prediction based on deep convolutional neural network. *SSRN Electronic Journal*. Retrieved from https://www.ssrn.com/abstract=3717738 doi: 10.2139/ssrn.3717738

Zhu, J., Chen, N., & Peng, W. (2019, 4). Estimation of Bearing Remaining Useful Life Based on Multiscale Convolutional Neural Network. *IEEE Transactions on Industrial Electronics*, *66*(4), 3208–3216. doi: 10.1109/TIE.2018.2844856

APPENDIX

A. DERIVING DVAE ELBO

For this derivation, the ELBO loss function is derived for a conditional DVAE that estimates the RUL sequences, $y_{1:T}$, given the sensor sequences, $x_{1:T}$, as conditional variables. To get the new ELBO loss for the conditional distribution, we can start with the negative log-likelihood of $p(y_{1:T}|x_{1:T})$. If this criterion is minimised, then it would be the same as maximising the likelihood samples $y_{1:T}$ belong to $p(y_{1:T}|x_{1:T})$, which is the aim of a generative model in the first place.

$$\begin{aligned} L &= -\log p(y_{1:T}|x_{1:T}) \\ &= -\log \int p(y_{1:T}, z_{1:T}|x_{1:T}) dz_{1:T} \end{aligned}$$

Note, $z_{1:T}$ is an arbitrary latent sequence we introduced to construct the generative model. One way to derive the ELBO is to use an importance distribution $q_\phi(z_{1:T}|y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})$ and use the same techniques found in importance sampling to approximate the more complicated distribution $p(y_{1:T}|x_{1:T})$. Normally, $q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})$ would be a simple distribution such as a Gaussian distribution.

$$\begin{aligned} L &= -\log \int p(y_{1:T}, z_{1:T}|x_{1:T}) \frac{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})}{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} dz_{1:T} \\ &= -\log \int \frac{p(y_{1:T}, z_{1:T}|x_{1:T})}{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} q_\phi(z_{1:T}|y_{1:T}, x_{1:T}) dz_{1:T} \\ &= -\log \mathbb{E}_{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \left[\frac{p(y_{1:T}, z_{1:T}|x_{1:T})}{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \right]. \end{aligned}$$

At this point, the expression is still too difficult to evaluate; normally, when deriving the ELBO for a VAE at this step, Jensen's inequality is used, $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$, where f is a convex function. Hence, this results in,

$$\begin{aligned} L &\leq -\mathbb{E}_{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \left[\log \frac{p(y_{1:T}, z_{1:T}|x_{1:T})}{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \right] \\ \mathcal{L} &= -\mathbb{E}_{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \\ &\quad \left[\log \prod_{t=1}^{T} \frac{p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})}{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \right] \\ &= -\mathbb{E}_{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})} \\ &\quad \left[\sum_{t=1}^{T} \log \frac{p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})}{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \right] \\ &= -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T}, x_{1:T})} \\ &\quad \left[\log \frac{p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})}{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \right]. \end{aligned}$$

Note, the last line uses the fact that expectations are a linear operator, i.e. $\mathbb{E}[f(x) + g(x)] = \mathbb{E}[f(x)] + \mathbb{E}[g(x)]$, and that expectations can cascade (Girin et al., 2021) i.e.,

$$\begin{aligned} &\mathbb{E}_{q_\phi(z_{1:T}|y_{1:T}, x_{1:T})}[f(z)] = \\ &\mathbb{E}_{q_\phi(z_1|y_{1:T}, x_{1:T})}[\mathbb{E}_{q_\phi(z_2|z_1, y_{1:T}, x_{1:T})}[\dots \\ &\mathbb{E}_{q_\phi(z_T|z_{T-1}, y_{1:T}, x_{1:T})}[f(z)] \dots]]. \end{aligned}$$

Expanding the expression further yields,

$$\begin{aligned} &= -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T}, x_{1:T})} \left[\log p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) \right] \\ &-\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T}, x_{1:T})} \left[\log \frac{p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})}{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \right] \end{aligned} \tag{31}$$

$$\begin{aligned} &= -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T}, x_{1:T})} \left[\log p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T}) \right] \\ &\quad + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t-1}|y_{1:T}, x_{1:T})} \\ &\left[\mathbb{E}_{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \left[\log \frac{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})}{p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})} \right] \right]. \end{aligned} \tag{32}$$

Note,

$$\begin{aligned} &\mathbb{E}_{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})} \left[\log \frac{q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})}{p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})} \right] \\ &= D_{KL} \left(q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T}) || p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T}) \right), \end{aligned}$$

which, therefore, gives the final expression for the DVAE ELBO loss function,

$$\mathcal{L} = -\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t}|y_{1:T},x_{1:T})} \left[\log p_{\theta_y}(y_t|y_{1:t-1}, z_{1:t}, x_{1:T})\right] + \sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_{1:t-1}|y_{1:T},x_{1:T})} \left[D_{KL}\left(q_\phi(z_t|z_{1:t-1}, y_{1:T}, x_{1:T})||p_{\theta_z}(z_t|z_{1:t-1}, y_{1:t-1}, x_{1:T})\right)\right]. \tag{33}$$

where Eq. 33 is the same as the ELBO stated in Eq. 15 in section 2.2.2.

B. HYPERPARAMETERS

The hyperparameters used to train the networks for each of the datasets are shown in Table 7 and Table 8 shows the hyperparameters for the baseline networks used in the Dust Filter experiment. When applying the Bayesian Optimisation scheme, the hyperparameter search was constricted to certain bounds for each hyperparameter. The bounds are stated for each hyperparameter in Table 9.

C. MODEL ARCHITECTURE

Table 10 shows how the Decoder, Encoder and Prior models are constructed for the DVAE model i.e. what layers are used and the input and output feature sizes used when specifying them in code.

Table 7. Hyperparameters chosen to train the DVAEs for each of the datasets in CMAPSS and the Dust Filter dataset

Hyperparameter	FD001	FD002	FD003	FD004	Dust Filter
learning rate	3.97×10^{-4}	3.20×10^{-4}	5.10×10^{-4}	6.65×10^{-4}	4.92×10^{-3}
Weight (L2) regularizer	5.80×10^{-6}	1.60×10^{-6}	1.10×10^{-6}	1.25×10^{-6}	4.00×10^{-6}
Batch size	393	246	215	196	467
Time Window (T)	49	44	34	48	20
Stride	1	2	1	2	1
Hidden dimension size	393	286	91	257	189
β	6.16	11.2	10.9	1.00	1.00

Table 8. Hyperparameters chosen to train the Baseline network models for the dust filter dataset. Note β is not applicable here as ELBO is not used for these relatively simple networks.

Hyperparameter	MLP	LSTM	GRU
learning rate	5.00×10^{-3}	1.10×10^{-3}	2.34×10^{-3}
Weight (L2) regularizer	2.15×10^{-6}	3.87×10^{-6}	9.97×10^{-6}
Batch size	150	228	273
Time Window (T)	38	39	41
Stride	5	1	1
Hidden dimension size	329	336	284

Table 9. Hyperparameter search bounds used during Bayesian Optimisation

Hyperparameter	Lower Bound	Upper Bound
learning rate	1×10^{-4}	5×10^{-3}
Weight (L2) regularizer	1×10^{-6}	1×10^{-4}
Batch size	150	550
Time Window (T)	20	50
Stride	1	5
Hidden dimension size	50	500
β	1.0	20.0

Table 10. Defines the networks based on sizes of outputs (ydim), inputs (xdim), latent variables (zdim) and the hidden variables to construct the hidden layers of the networks (hdim). Note the inputs to the network have hdim because of the encoded sequences $\mathbf{x}_{1:T}$ and $y_{1:T}$. For example, the Encoder network has hdim$\times 2$ as the input size because it takes both $\mathbf{x}_{1:T}$ and $y_{1:T}$ as inputs and these are encoded with the RNN encoder so their embedded representation has dimension hdim.

Layer	Input feature size	Output feature size
	Decoder	
Linear	hdim+zdim	hdim
tanh	hdim	hdim
Linear	hdim	hdim
tanh	hdim	hdim
Linear	hdim	ydim$\times 2$
	Prior	
Linear	hdim+zdim	hdim
tanh	hdim	hdim
Linear	hdim	hdim
tanh	hdim	hdim
Linear	hdim	zdim$\times 2$
	Encoder	
GRU	hdim$\times 2$	hdim
Linear	hdim	hdim
tanh	hdim	hdim
Linear	hdim	hdim
tanh	hdim	hdim
Linear	hdim	zdim$\times 2$
	$\mathbf{x}_{1:T}$ **or** $y_{1:T}$ **Encoder**	
GRU	xdim or ydim	hdim
GRU	hdim+(xdim or ydim)	hdim