UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

BAYESIAN RECURRENT NEURAL NETWORKS FOR DIAGNOSIS OF FAULT MODES AND PROGNOSIS OF REMAINING USEFUL LIFE

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

JOSÉ MANUEL CÁCERES VALENZUELA

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
RODRIGO PASCUAL JIMÉNEZ

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS
DE INGENIERÍA MECÁNICA
POR: **JOSÉ MANUEL CÁCERES VALENZUELA**
FECHA: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

BAYESIAN RECURRENT NEURAL NETWORKS FOR DIAGNOSIS OF FAULT MODES AND PROGNOSIS OF REMAINING USEFUL LIFE

Deep learning has emerged as a promising method to handle big amounts of machinery data and use it to determine the remaining useful life or the health state in the context of prognostics and health management. However, most recent studies represent the results of neural networks in form of a single point, and thus cannot properly represent the uncertainties in predictions. This practice usually provides overly confident predictions and misguide the decision-maker that might cause severe consequences in safety-critical situations.

This work proposes to integrate variational inference and recurrent neural networks to quantify and propagate uncertainty in fault diagnosis and prognosis models. This exploits the recent progress in weight perturbations to represent the weights of a neural network in form of a distribution and introduce a regularization effect on the network. The proposed approach is validated in a simulated dataset (CMAPSS) to check its performance against different models, finally, the proposed approach is tested on four datasets obtained experimentally to determine its ability to diagnose fault modes and forecast the remaining useful life time in different mechanical equipment.

The results obtained show that the proposed approach is capable of carrying out diagnostic and prognostic tasks, presenting outstanding results, in addition to propagating and quantifying uncertainty.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS
DE INGENIERÍA MECÁNICA
POR: **JOSÉ MANUEL CÁCERES VALENZUELA**
FECHA: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

BAYESIAN RECURRENT NEURAL NETWORKS FOR DIAGNOSIS OF FAULT MODES AND PROGNOSIS OF REMAINING USEFUL LIFE

El aprendizaje profundo se ha convertido en un método prometedor para manejar grandes cantidades de datos de maquinaria y usarlo para determinar la vida útil restante o el estado de salud en el contexto de pronósticos y diagnóstico de fallas. Sin embargo, los estudios más recientes representan los resultados de las redes neuronales en forma de un solo punto, y por lo tanto no pueden representar adecuadamente la incertidumbre. Esta práctica generalmente proporciona predicciones demasiado confiables y desorienta al momento de tomar decisiones, lo que podría causar graves consecuencias en situaciones críticas de seguridad.

Este trabajo propone integrar inferencia variacional y redes neuronales recurrentes para cuantificar y propagar la incertidumbre en los modelos de diagnóstico y pronóstico de fallas. Para esto, se utilizan los recientes progresos en el area de *weight perturbations* para representar los pesos de una red neuronal en forma de distribución e introducir un efecto de regularización en la red. El enfoque propuesto se valida en un conjunto de datos simulados (CMAPSS) para verificar su rendimiento con respecto a diferentes modelos, finalmente, el enfoque propuesto se prueba en cuatro conjuntos de datos obtenidos experimentalmente para determinar su capacidad de diagnosticar modos de falla y pronosticar el tiempo de vida útil restante en diferentes equipos mecánicos.

Los resultados obtenidos muestran que el enfoque propuesto es capaz de llevar a cabo tareas de diagnóstico y pronóstico, presentando resultados sobresalientes, además de propagar y cuantificar la incertidumbre.

Do. Or do not.
There is no try.

Acknowledgments

En primer lugar, agradecer a mis padres Teresa y Cesar, que siempre ha estado apoyándome de forma incondicional en las decisiones que me trajeron hasta aquí, nunca podre agradecerles todo lo que significo para mi que a pesar de todo siempre pude contar con ustedes, junto a ustedes también agradezco a mi hermano Cesar, cuantas peleas habremos tenido, sin embargo, para cuando de verdad cuenta, siempre estas para apoyarme y aunque no lo diga siempre, no podría estar más agradecido por el hermano mayor que tengo.

A mis amigos de la vida, nunca olvidare todas las estupideces y aventuras que pasé con quienes hasta el día de hoy son mis amigos, Claudio, Cristian, Juanito, Víctor, Felipe, Yerko, Sergio, Matías y Matías-Pedro, quienes a pesar de todos estos años aún siguen ahí. A mis amigos de la Universidad, a pesar de considerarme a mi mismo alguien un tanto desagradable, siempre sentí que podía pasar un buen momento compartiendo con todos los buenos amigos que hice en la u, nunca olvidare todas las idas en la 506 con Carlitos y Papo hablando del futuro y de las copuchas de la U, más generalmente con Papo pues Carlos no siempre estaba en un estado adecuado para conversar, siempre disfrute la forma en que la Cami me sermoneaba por ser tan despreocupado y a veces poco atento o simplemente pesado, tampoco olvidare todas las tardes de conversaciones con la Noe, Tente, Italo, Panda y el Coto que trataban de despejarse de la U hablando de otros temas que siempre terminaban en la U.

También agradecer de forma especial a dos amigos que hicieron que mi estadía en la U fuera de las mejores experiencias en mi vida, Joaquín a quien conocí en primer día , el crack incondicional adicto al animé y a Pokémon que siempre ha estado para mí en las buenas y en las malas, mi segundo amigo es el crack Iván, una amistad que comenzó en mis primeros años en el departamento de ingeniería mecánica, a pesar de no hablar siempre, se que puedo contar contigo, nunca olvidare las noches que íbamos a mi casa en Maipú a estudiar o hacer algún trabajo para el día siguiente o cuando nos quedamos a estudiar en la u toda la noche para el examen de máquinas, incluso nos acompañamos en el magister, gracias a ambos por todos los buenos momentos que viví gracias a ustedes, son lo más grande.

Finalmente agradecer a la persona que día a día me ayuda a ser una mejor persona, Meli has sido un apoyo incondicional para mí a pesar de que a veces sea difícil, siempre me ayudas y sabes que hacer para que cada día sea una experiencia nueva llena de alegría y sorpresas, ni con todas las palabras de este podría expresar lo feliz que me haces, pero sé que cada día puedo comenzar con un te amo, junto a ti, también agradecer al Doggo más regalón de todo el universo, nuestro ratón Tommy que siempre esta con nosotros y que nos ama de la forma mas pura e incondicional, nunca podría hacerle el nanai necesario para que sepa cuanto lo amo.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The modern engineered systems are generally configured by individual components, humans and their interactions. This characterization exponentially increases the complexity of systems, in turn, boosting the possibility of system failure and therefore high system risk. Defining and understanding the behavior and interactions among systems becomes challenging and even impossible. With the rapid development of the Internet of Things (IoT), these modern systems tend to be increasingly instrumented with network-connected devices and massive quantities of multidimensional data is constantly generated. These big machinery data has been recognized as the most valuable resource, based on which machine learning techniques can be utilized to discover the hidden features to gain better insights of the system performance [23].

In the past decade, prognostics and health management (PHM) has attracted increasing attention from both academia and industry. PHM uses sensor technology and data analytics to detect the degradation of engineered systems, diagnose the type of faults, predict the remaining useful lifetime (RUL) and proactively manage failures. Typically, the raw data is first processed to enhance the data quality, then identify the fault relevant features and use machine learning techniques to formulate a predictive model for diagnostic and prognostic purposes. Most recent PHM studies focus on the critical machinery components including bearings [37, 43], gears [26, 38] and batteries [16, 29]. However, these studies are mainly developed based on conventional machine learning with shallow configuration and thus, large amounts of time and expertise resources to manually design features.

More recently, deep learning has emerged as a promising solution that allows to handle big machinery data and automatic learning from data representing features without deep knowledge about it. The key idea of deep learning is to use multiple layers to progressively learn a more abstract and complex representation of the raw input, and ultimately formulate an end-to-end predictive framework. Typically, there are several types of deep learning models including Auto-encoder, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network and Deep Boltzmann Machines. Variants of these models are also available and are still under active development such as Variational Auto-encoder, Generative adversarial networks and long short-term memory (LSTM). In the context of machinery health prognosis, large amounts of research efforts have been conducted to explore the advantages against conventional machine learning techniques [17]. Verstraete et al. developed an unsupervised and a semi-supervised deep learning generative adversarial network-based methodology for fault diagnostics of rolling element bearings [41]. Zhang et al. presented a LSTM RNN model to predict the remaining useful life of lithium-ion batteries [48]. Correa-Jullian et al. [7] demonstrated that the deep learning models for solar thermal systems can outperform predictors and other regression models such as Bayesian Ridge, Gaussian Process and Linear Regression. Two excellent review articles on the applications of deep learning to handle big machinery data are developed by Zhao and Jia et al. [49, 19].

It should be noticed that the deep learning models developed in the mentioned studies are unable to determine the uncertainty present in the operational data of the system and in the model. Specifically, the weights of the deep learning models are represented in form of a single point estimation and thus, cannot properly represent their uncertainties in predictions. This practice usually provides overly confident predictions and then misguide the decision-maker that might cause severe consequences in safety-critical industries. Although Peng et al. [33] proposed a Bayesian deep learning approach to address uncertainty through health prognosis, their Bayesian approximation is implemented through Monte Carlo (MC) Dropout that unlike the regular dropout, MC Dropout is applied at both training and test. Then, the addition of dropout between every layer can turn some portion of neurons in each layer to 0, returning the output without some extracted features by the network. This ultimately generates random predictions as samples from a probability distribution which is treated as variational inference [14]. However, the performance of MC Dropout has been criticized by Osband et al. [30, 32], and counterexamples were presented to demonstrate the limitations of MC Dropout to produce satisfactory results.

Nowadays, Bayesian recurrent neural networks has been studied and developed in different ways: [28] using a Markov Chain Monte Carlo variation (PX-MCMC) or [13] using Bayes By Backprop with reparametrization trick to estimate the uncertainty of the model. As is shown in Wen et al. [44], the flipout method outperforms previous methods using Bayesian neural networks, because of this, in this work, an implementation of the most simple recurrent neural network in their Bayesian form using flipout approach is developed and validated in different datasets.

In particular, the weights of a Bayesian neural network are represented in the form of a distribution rather than a single point estimation. As such, the model uncertainties can be considered through the Bayesian posterior inference over the weights of the Bayesian recurrent neural network. Meanwhile, a regularization effect is introduced by specifying a prior distribution for the weights of a neural network. The Bayesian approximation is conducted through a pure variational inference with Bayes by backprop. The proposed approach is validated using an open-access turbofan engine dataset [36] that is the most usual benchmark for RUL prognosis. After the models are validated in CMAPSS dataset, they are tested in four datasets more, to prove their efficiency in real life datasets.

The remainder of this paper is organized as follows. Chapter 2 provides a brief summary of RNNs and their well-known architectures VRNN and JANET. Also, the background of Variational inference, Bayes by Backprop and weight perturbation methods. Chapter 4 presents the proposed Bayesian RNN approach including the equations that characterizes the Bayesian algorithm, a flowchart and workflow of this work approach and the architectures that are compared with other models. Section 5 briefly summarizes all the datasets, all their preproccesing and discusses the results using different models. Finally, chapter 6 provides conclusions and recommendations.

1.1. Hypotheses

The first hypotheses of this thesis is that Bayesian recurrent neural networks are competent in using the temporality of the data to estimate the remaining useful life of a mechanic system or its health state.

The second hypotheses is that Bayesian recurrent neural networks are capable of estimating and characterizing the prediction uncertainty in terms of a probability distribution.

The last hypothesis is that obtaining the uncertainty of the predictions, the results obtained are better than in frequentist neural networks in terms of RMSE for prognosis and accuracy for diagnosis.

1.2. General Objectives

Develop and implement two models of recurrent Bayesian neural networks and show their effectiveness in the context of predictive maintenance, specifically in the tasks of predicting remaining useful life and diagnosing health status.

1.3. Specific Objectives

1. Show a detailed mathematical development and discussion of the proposed Bayesian Recurrent Neural.
2. Examine and implement the most used architectures of frequentist recurrent neural networks in their Bayesian formulation.
3. Test the model developed in different datasets to demonstrate its generalization skills in diagnosing failure modes and prognosis of RUL.

1.4. Resources Available for this Thesis

For the computational experiments required for the realization of this work, the SRMI Lab of University of Chile facilitated the following hardware resources:

- Desktop computer with processor Intel(R) Core(TM) i7-8700 CPU @ 3.70 GHz, Windows 10 Pro as operative system, 32 GB of DDR4 RAM and a GPU Nvidia Titan Xp.

The software resources were open sources libraries which include:

- Python 3.6 programming language.

- Tensorflow 1.13
- Tensorflow probability
- Pandas, Numpy, Seaborn, Matplotlib, SKlearn were used to manage the data.

Chapter 2

Theoretical Background

The following chapter shows the background necessary to understand the model developed in this work. In the first instance, the context within which the project is developed is revealed, this is preventive health management. Then, the relevant scientific areas for development are presented, which are machine learning and a particular subset of it: deep learning in which the most common algorithms for solving classification and regression problems are illustrated, and finally, the variational inference method is introduced to solve problems that involve the determination of probability distributions.

2.1. System-Level PHM definition

Mechanical systems such as pumps, turbines, bearings, among others, provide services that are essential for nowdays societies. These systems are completely monitored by different types of sensors and the information obtained from these sensors is used to predict the time of failure of the equipment or their state of health. PHM is the study of advanced techniques for the analysis of this data and through different techniques it will be able to automatically determine the condition of some equipment[9], however, the commonly used methods are not capable of delivering completely robust results, this means, they cannot provide a measurement of the uncertainty present in them, so the interpretation of the results is limited.

To tackle the aforementioned challenge. it is imperative to use models able to measure the uncertainty present in themselves, this allows decisions to be made with more information on the real state of some equipment and thus to carry out preventive measures with greater reliability.

2.2. Machine Learning

Machine learning is a method of data analysis based on the idea that systems are capable of learning from data to recognize patterns with a high degree of automation. The type of problems that can be solved with machine learning techniques can be divided in four categories: classification, regression, clustering and anomaly detection[3]. In the PHM context, the most typically requested tasks are those of classification and regression. Classification tasks in this context are mainly based on the identification of the state of operation of various mechanical systems or the detection of anomalies in these, while regression tasks are commonly used in the context of determining the remaining useful life of some equipment or to predict any index related to the useful life of this. The regression and classification problems are presented below.

2.2.1. Classification and Regression Problems

The objective in the classification tasks is to teach some algorithm to separate different input values into different categories, that is to say, the problem is to relate the input data x in the different categories C, for this, the algorithm used must be able to extract the most significant characteristics from the data and use them for their categorization. This thesis focuses on categorizing temporary data within a certain state of health corresponding to the mechanical equipment.

Unlike classification problems, regression problems focus on predicting a value that, in the context of this study, is capable of determining the health status of a team. Basically, the main task of a regression algorithm is to find the function $y = f(x)$, where y is the indicator of the health status of the equipment and f is the function that the model must approximate, commonly this type of problem in the context of PHM is more complex than the type of classification problems, this is because there is not always a direct relationship between the failure time of an equipment with its operational state, which increases the complexity of this type of problem.

Machine Learning problems can also be divided based on the availability of data in two distinct categories: supervised learning and unsupervised learning [15]. This work is focused in supervised learning that is explained below.

2.2.2. Supervised Learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs[35].This means that the supervised models are designed to understand the characteristics necessaries to generate a mapping between x and y. Typically, regression and classification problems are also supervised problems.

In PHM, most problems involve supervised learning, since the operating status of a mechanical equipment is generally known while it is monitored.

2.3. Deep Learning

Deep learning is a class of machine learning algorithms [10] that uses multiple layers to extract higher level features from the input data. In this way, it is possible to get more information of the same data. Neural networks purpose is to approximate a function based in large amount of data, these types of models are composed by layers or cleverly ordered parameters trained using a technique based on gradient descent called automatic differentiation [22]. There are a lot of tasks neural networks can easily perform such as regression, classification and data reconstruction.

This section reviews three types of neural networks architecture that are used in this thesis: artificial neural networks (ANN), convolutional neural networks (CNN) and recurrent neural networks (RNN).

2.3.1. Artificial Neural Networks

Artificial Neural Networks (ANN), are the basis blocks of the deep learning models. They consist of sets of neurons in multiple hidden layers that operate the input data of each one of them. This models ends with the output layer that predicts the target of the model. A simple scheme of a ANN is presented in Figure 2.1

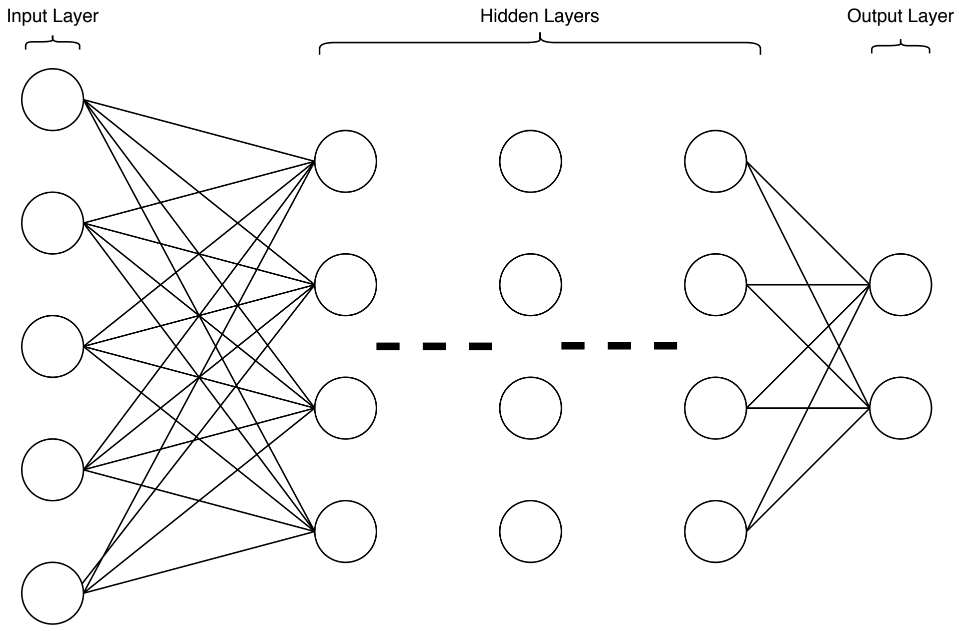


Figure 2.1: ANN scheme.

The ANN's apply a linear operation followed by a nonlinear function σ known as activation function which is presented in equation 2.1.

$$\vec{y} = \sigma(W\vec{x} + \vec{b}) \tag{2.1}$$

Where matrix W and vector $\vec{b}$ are parameters known as *weights* and *biases* respectively, those are parameters that are trained using the input data of the model. Commonly, the deep learning architectures stack large amount of layers with the aim of extract deeper features that can be useful to complete a different task.

2.3.2. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are an ANN-based model that uses a receptive field named *kernel K* to extract patterns from the input data through multiple stacked *convolutional layers*, each neuron in the convolutional layers is connected only to neurons located within the *kernel* in the previous layer.

To do the convolution operation ($\star$) is used within a matrix of weights W_k (that represents the kernel K) and the input data $X \in R^{IxJ}$ from an arbitrary layer as shown in equation 2.2.

$$y_{(i,j)} = (W_k \star W) = \sum_{a=0}^{w} \sum_{b=0}^{h} W_{k(a,b)} \cdot X_{i+a,j+b} \tag{2.2}$$

where $y_{(i,j)}$ corresponds to the output of the convolution of the kernel K, also named feature map, ww and h are the width and height of K respectively and (i, j) corresponds to and specific point in X. With this, the convolutional layers are able to generate multiple feature maps to interpret the information of the input data.

Finally using equation 2.2, the output of a convolutional layer can be written as is shown in equation 2.3.

$$\vec{y} = \sigma(W_k \star X + \vec{b}) \tag{2.3}$$

An schematic view of a common CNN is presented in Figure 2.2. It can be seen that the output of the last convolutional layer is flattened to be used as input of the ANN and perform the desired task, in this case, regression.

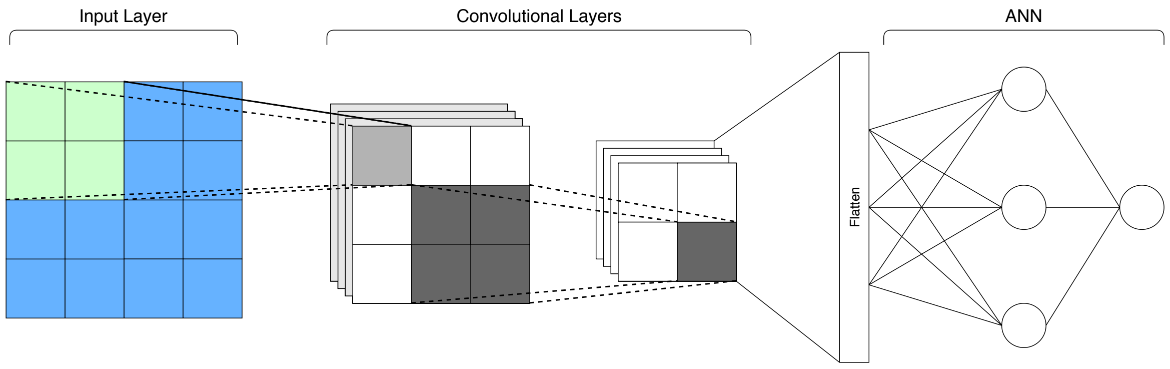


Figure 2.2: CNN scheme.

Convolutional neural networks are widely used because of their capacity to generalize and extract information of the input dataset, in fact, the CNN's outperform common ANN's in experiments of pattern recognition and automatic feature extraction, therefore, these networks are widely used in PHM[21].

2.3.3. Recurrent Neural Networks

Recurrent neural networks(RNN) are artificial neural networks capable of operating time series, that is to say, they can deal with data in the form $x = \vec{x_1}, \ldots, \vec{x_t}$ [15], this type of data is quite common in the context of PHM, since the monitoring of most mechanical equipment is done using temporary sensors such as accelerometers, speedometers, tachometers, etc. All recurrent neural networks are made up of basic units known as *cells*, which can be presented as single units or as a group. For these architectures it is necessary to take advantage of one of the first ideas found in machine models learning and statistics from the 1980s: sharing parameters through the different parts of the same model, in this case, the input data passes through the model in a sequential way so that a single output data is obtained for each time-step, this output data is known as hidden state. This hidden state also serves as an extra input for the next time-step, alongside the actual input for that time-step, so the cell has information about the last result predicted and the data of the current time-step. With this information, the cell understands the temporal behavior of the data, so it can produce an output to the actual time-step according to that development. This process is schematized by a basic RNN in Figure 2.3, referred to as vanilla RNN (VRNN), in which the network of neuron-like nodes organizes into successive layers, the general operation for vanilla RNN is shown in equation 2.4:

$$h_t = \sigma(U \cdot x_t + V \cdot h_{t-1} + b_h) o_t = h_t$$

Where x_t denotes the input data at time t, h_t denotes the hidden state of the cell in time t and o_t denotes the output of the cell in every time t.

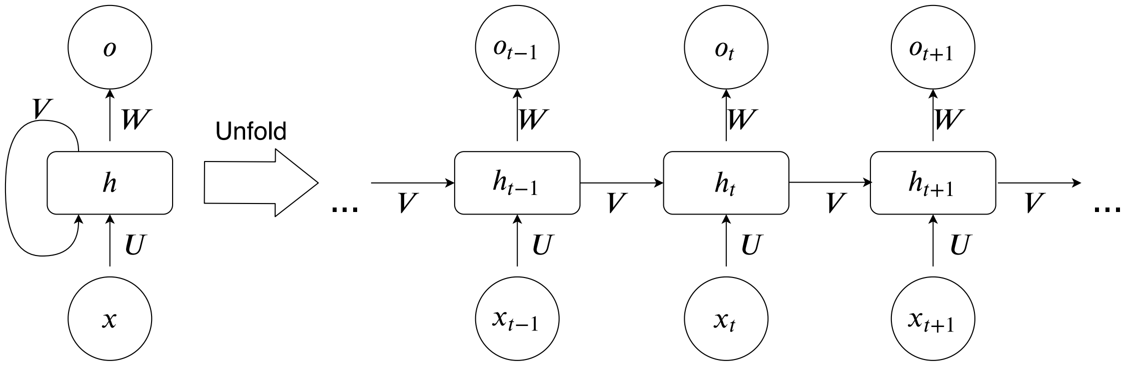


Figure 2.3: Architecture of a RNN.

In the following of this section, two of the most well-known cells in recurrent neural networks are studied: Vanilla recurrent neural networks(VRNN) and just another network (JaNet).

- Vanilla Recurrent Neural Network (VRNN):
  Basic VRNN is a network of neuron-like nodes organized into successive *"layers"*. Each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer.
  The VRNN architecture is basically composed of one hidden gate, as is shown in Figure 2.4, the activation function of this gate is normally a hyperbolic tangent.

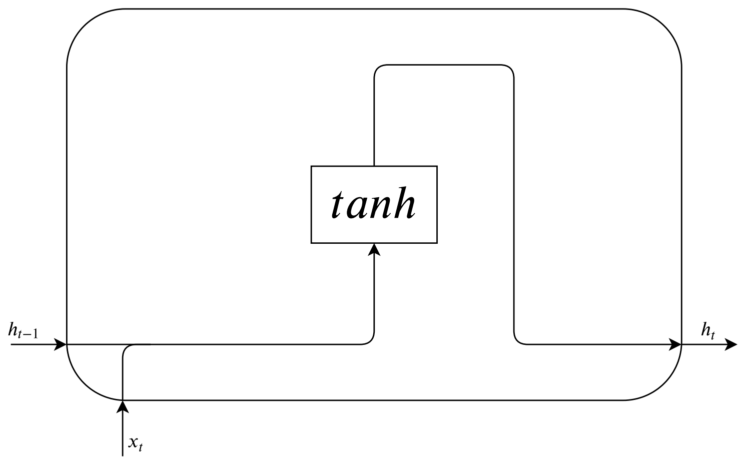


Figure 2.4: RNN cell.

The equation below describes the operation inside the cell [15]:

$$\begin{aligned} h_t &= tanh(U \cdot x_t + V \cdot h_{t-1} + b_h) \\ o_t &= h_t \end{aligned} \tag{2.4}$$

Where:

- U_h: are the weight matrix of each of the layer for their connection to the current input vector x_t.
- V_h: are the weight matrix of each of the layer for their connection to the hidden state vector h_{t-1}.
- b_h: are the bias terms for each of the layers.

▪ Just Another Network (JaNet):
Due to the new perspective that Gated recurrent networks [6] implemented (less gates, fast training, best results), it is natural to ask whether gates are completely necessary or not. The JANET Cell only uses a forget cell (f_t). As shown by Westhuizen et al. [39], the JANET architecture can not only perform better than the common LSTM network on the MNIST and pMNIST dataset, but also require less computational resources. The basic JANET cell is shown in Figure 2.5.

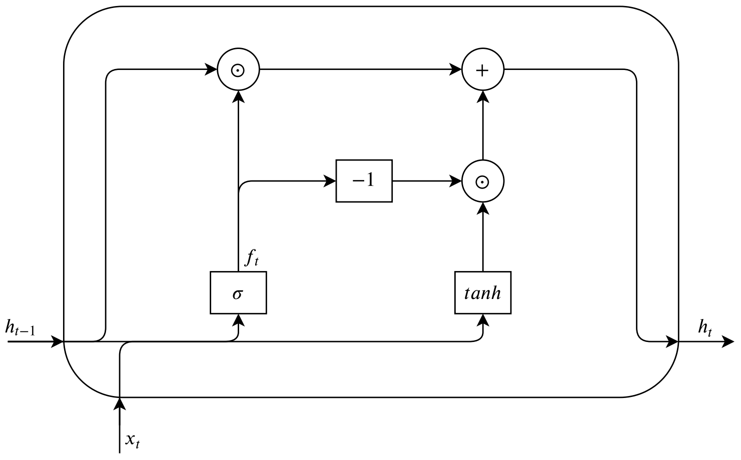


Figure 2.5: JaNet cell.

The JANETs operations are described below: [39]:

$$f_t = \sigma(U_f \circ h_{t-1} + W_f \cdot x_t + b_f)$$

$$c_t = f_t \circ (1 - f_c) \circ tanh(U_c \cdot h_{t-1} + W_c \cdot x_t + b_c) \tag{2.5}$$

$$h_t = c_t \tag{2.6}$$

Where:

- W_i: are the weight matrix of each of the layer for their connection to the current input vector x_t.
- U_i: are the weight matrix of each of the layer for their connection to the hidden state vector h_{t-1}.
- b_i: are the bias terms for each of the layers.
- σ: is the activation funcion.

In the next section a review of the training procedure of all the networks previously mentioned is presented.

2.3.4. Training of Neural Networks

As it was show in the previous section, all the purpose of neural networks is to produce a function f with parameters θ (all the trainable parameters) able to estimate certain values as is shown in equation 2.7.

$$\hat{y} = f(x) \tag{2.7}$$

where x is the input of the network that can be a vector of data, images or sequence of vectors depending on the type of network, and $\hat{y}$ are the predicted values. In supervised learning, the data consists in pair of points (x, y) where x is the data, y are the labels of this data, and the parameters θ are learned by minimizing a function denominated *loss function* L that is selected based on the type of task performed by the network, but the mainly objective of the network is that the predicted values $\hat{y}$ are as similar as possible to the labels y for each data x.

The process of optimization is done firstly by computing the gradient of each parameter in θ in relation to the loss function L, this is, $\frac{dL}{d\theta}$ with backpropagation algorithm and using gradient descent to update each parameter in θ. This process is faster and more stable when the gradient descent algorithm is applied to an averaged sample subset of the dataset to update the weights and biases. The size of the subsample is known as batch size. When every pair of the original dataset has been used to compute gradients at least once, it is said that one epoch of training has passed.

For regression problems, the value $\hat{y}$ that the net has to predict is just one, which means that the last layer of the neural networks only has one neuron and doesn't have an activation function, i.e the activation function is the identity function. In this case, the metric to evaluate the performance of the model is basically the distance between the real value y and the predicted values $\hat{y}$, as the distance is a positive value, the way to avoid problems with negative values is to get the quadratic distance. Lastly, to avoid a dependence with the data size, the total sum of the distances is divided for the data size N as is shown in equation 2.8, this value is known as*mean squared error*(MSE).

$$MSE = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.8}$$

Another loss function commonly used for regression task is the *root mean squared error*(RMSE), this is because taking the square root of the average squared errors has some interesting implications. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable, the equation that describes RMSE is showed below 2.9.

$$RMSE = \sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.9}$$

For the classification problems, the output of the neural network is equal to the number or classes in the problem (C), which means that the last layer of the net has the same number of neurons that the classes in the problem, when the problem is a multi-class problem, the activation function that correspond to the last layer is the softmax function[17], presented in equation 2.10. Softmax is a function that takes as input a vector of C real numbers, and normalizes it into a probability distribution consisting of C probabilities proportional to the exponentials of the input numbers.

$$\sigma(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{C} e^{\hat{y}_j}} \tag{2.10}$$

A Scheme for the training process of an ANN is presented below in figure 2.6:

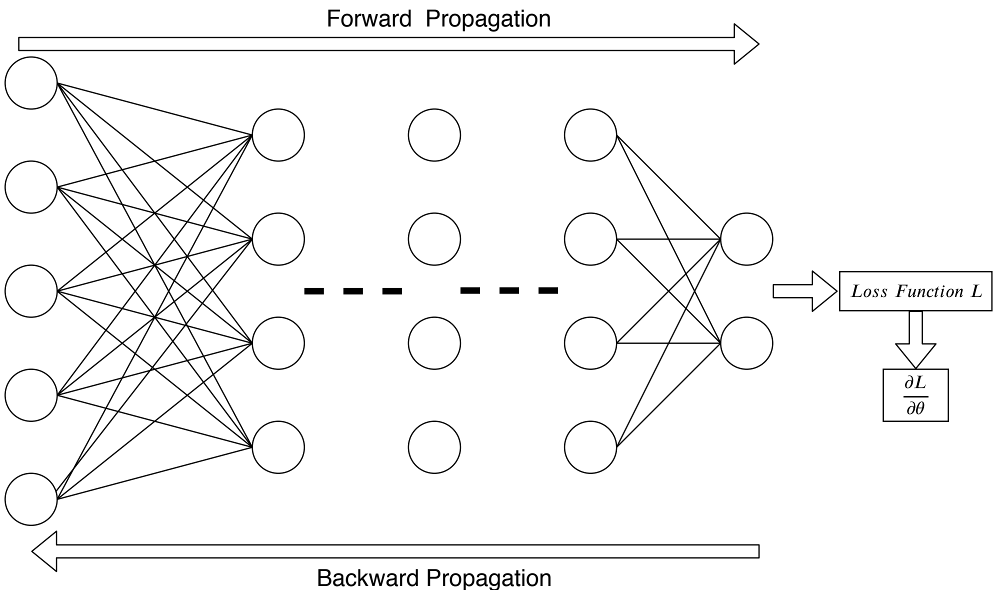


Figure 2.6: Schheme of the training process in ANN's

The following section explains a technique for approximating probability distributions called *variational inference*, this technique is essential for the development of the main model of this thesis.

2.4. Variational inference

One of the main current statistical problems corresponds to the ability to adequately process probability densities. To face this type of problem, the variational inference tool was developed, a method mainly dedicated to transform the problems of finding probability distributions into problems of optimization [20]. To understand the use of this tool, let's examine the following example: consider $x = x_1, \cdots, x_N$ the set of observable variables and $z = z_1, \cdots, z_N$ a set of latent variables, in this case, the latent variables are the representation of the weights of the model sampled from a distribution of parameters θ, the inference problem is to compute the conditional density of the weights given the observations, $p(z|x)$, then, the expression of the conditional density can be written as shown in equation 2.11.

$$p(z|x) = \frac{p(z, x)}{p(x)} \tag{2.11}$$

The denominator of equation 2.11 is denoted as *evidence* and can be calculated by integrating the latent variables of the joint density 2.12. The marginalization over z to calculate $p(z)$ in the denominator is typically intractable, because, for example, the search space of z is combinatorially large. For almost every model this integral is unavailable in closed form, and the evidence is what we need to compute the conditional from the joint distribution.

$$p(x) = \int p(x, z)dz \tag{2.12}$$

In variational inference, a family Q of densities over the latent variables is specified. Each $q \sim Q$ is a candidate approximation to the exact conditional distribution, the final goal is to find the distribution q closest to the joint distribution $(p(z|x))$ [4], this is, the distribution that minimize the Kullback–Leibler divergence. This transforms the problem of the equation 2.11 and 2.12 in the following problem:

$$q^*(z|\theta) = arg\ min_{q(z|\theta)\sim Q}\ KL(q(z|\theta)||p(z|x)) \tag{2.13}$$

where q^* is the best approximation to solve the optimization problem, however, this is still not computable due to the problem in equation 2.12. To understand why, it is necessary to work a little in the KL divergence, for this, the Expectation in relation to a distribution and the Kullback-Leibler divergence are defined as:

$$\mathbb{E}_{q(z|\theta)} = \int q(z|\theta)f(z)dw \tag{2.14}$$

$$KL(q(x)||p(x)) = \int q(x)log\ \frac{q(x)}{p(x)}dx \tag{2.15}$$

Using this in the original problem of equation 2.13 the following equation is obtained:

$$KL(q(z|\theta)||p(z|x)) = \mathbb{E}_{q(z|\theta)\sim Q}[log\ q(z|\theta) - log\ p(z|x)] \tag{2.16}$$

Expanding the conditional using Bayes theorem it results as:

$$KL(q(z|\theta)||p(z|x)) = \mathbb{E}_{q(z|\theta)\sim Q}[log\ q(z|\theta) - log\ p(x|z) - log\ p(z) + log\ p(x)] \tag{2.17}$$

As $p(x)$ is a value that is not a probabilistic function, the expected value $(\mathbb{E}_{q(z|\theta)\sim Q}[log\ p(x)])$ is equal to $p(x)$:

$$KL(q(z|\theta)||p(z|x)) = \mathbb{E}_{q(z|\theta)\sim Q}[log\ q(z|\theta)] - log\ p(x|z) - log\ p(z)] + log\ p(x) \tag{2.18}$$

As this reveals the dependency of $p(x)$ which is the problem in the fist place, now, to solve this issue it's necessary to work even more in the equation 2.18, regrouping some terms it results in:

$$KL(q(z|\theta)||p(z|x)) - log\ p(x) = \mathbb{E}_{q(z|\theta)\sim Q}[log\ q(z|\theta) - log\ p(x|z) - log\ p(z)]$$

$$KL(q(z|\theta)||p(z|x)) - log\ p(x) = \mathbb{E}_{q(z|\theta)\sim Q}[log\ q(z|\theta) - log\ p(z)] - \mathbb{E}_{q(z|\theta)\sim Q}[log\ p(x|z)]$$

$$KL(q(z|\theta)||p(z|x)) - log\ p(x) = KL(q(z|\theta)||p(z)) - \mathbb{E}_{q(z|\theta)\sim Q}[log\ p(x|z)]$$

$$log\ p(x) - KL(q(z|\theta)||p(z|x)) = \mathbb{E}_{q(z)\sim Q}[log\ p(x|z)] - KL(q(z|\theta)||p(z)) \tag{2.19}$$

The main objective of this was to minimize the Kullback–Leibler divergence on the left side of the equation 2.19, which minimizes the *distance* between the proposed distribution

$q(z)$ and the real posterior function $p(z|x)$. If only the proposed distribution is changed, it will not variate the first term in the left side of the equation 2.19 because the *evidence* just depends of the data, therefore, if the right side of the equation is maximized by varying $q(z|\theta)$ the Kullback–Leibler divergence on the left side will automatically decrease. The right side of the equation is named the Evidence Lower Bound (ELBO), which derives from it being always a lower bound for $log\ p(x)$ (the evidence) since the Kullback-Leibler divergence is always a non-negative quantity [4], the equation for ELBO is seen in equation 2.20.

$$-ELBO = KL(q(z|\theta)||p(z)) - \mathbb{E}_{q(z|\theta)\sim Q}[log\ p(x|z)] \tag{2.20}$$

This equation is one of the most important in Bayesian problem under Deep Learning context so it is important to understand each term of the equation. The second term describes the probability of the data given the latent variables; it encourages densities that place their mass on configurations of the latent variables that explain the observed data. The first term is the negative divergence between the variational density and the prior; it encourages densities close to the prior. Thus, the variational objective mirrors the usual balance between likelihood and prior.

Summarizing, Maximizing the ELBO is equivalent to minimizing the Kullback-Leibler divergence, for the most part, it is easier to treat the maximization problems as a minimization one, therefore, instead of maximizing the ELBO, it is preferred to minimize the negative value of it.

2.4.1. Weight Perturbations

Weight perturbations is a class of methods that instead of use weights as a collection of numbers, uses weights as a collection of values sampled from a distribution q_θ parametrized by θ. The objective of this methods is to minimize the expected loss function, the distribution q_θ can often be described in terms of perturbations: $W = \bar{W} + \Delta W$, where W are the mean weights and ΔW is a stochastic perturbation.

Some specific examples of weight perturbations are Gaussian perturbations, Dropconnect [42], variational Bayesian neural nets and Evolution strategies[34].

In general, weight perturbation algorithm suffers high variance of the gradient estimates because all training examples in a mini-batch share the same perturbation, which means that sharing perturbations make the net *think* that there is a relation between the weights, this causes lack of precision and the variance can't be eliminated by averaging. The method of flipout is presented below.

- **Flipout**:

  To apply Flipout method is necessary to make two important and non-trivial assumptions over q_θ:

  1. The perturbation over the weights are independent.

2. The perturbation distribution is symmetric around zero.

Is important to note that under this assumptions, the perturbation distribution is invariant to element-wise multiplication by random sign matrix (a matrix whose entries are ± 1). From now on, element-wise multiplication is denoted by $\circ$.

Before explaining mathematically this method, it is necessary to understand the following observation: *Let q_θ be a perturbation distribution that satisfies the above assumptions, and let $\Delta W \sim q_\theta$. If E is a random sign matrix independent of $\Delta\hat{W}$, then $\Delta W = \Delta\hat{W} \circ E$ is identically distributed to ΔW. Furthermore, the loss gradients computed using ΔW are identically distributed to those computed using $\Delta\hat{W}$.* [44]

Flipout method uses this fact by using an *original perturbation* $\Delta\hat{W}$ shared by all the data points in the batch and multiplies it by different rank-one sign matrix as is shown below:

$$\Delta W_i = \Delta\hat{W} \circ r_i s_i^T \tag{2.21}$$

where the subscript denotes the index within the mini-batch, and r_i and s_i are random vectors whose entries are sampled uniformly from ± 1. Using this "trick", the marginal distribution over gradients computed for individual training examples will be identical to the distribution computed using shared weight perturbations. Consequently, flipout yields an unbiased estimator for the loss gradients, the next step can be described as shown in equation 2.4.1:

$$\begin{aligned} y_i &= \sigma(W^T x_i) \\ y_i &= \sigma((\bar{W} + \Delta\hat{W} \circ r_i s_i^T)^T x_i) \\ y_i &= \sigma(\bar{W}^T x_i + (\Delta\hat{W}^T (x_i \circ s_i)) \circ r_i) \end{aligned} \tag{2.22}$$

To vectorize the previous equation it is necessary to define matrices R and S whose rows correspond to the random sign vectors r_i and s_i, then the equation 2.4.1 can be written as equation 2.23

$$Y = \sigma(X\bar{W} + ((X \circ S)\Delta\hat{W}) \circ R) \tag{2.23}$$

This defines the next step because R and S are sampled independently of W and $\Delta\hat{W}$, a backpropagation of the gradients through equation 2.23 can be done to obtain derivatives with respect to W, $\Delta\hat{W}$, and X.

Summarizing, flipout is a weight perturbation method able to regularize neural networks and decorrelate the weight gradients between different examples in a batch.

Also, Flipout takes advantage of evolution strategies allowing parallelism on a GPU updating the weights with the following scheme:

$$\bar{W_{t-1}} = \bar{W}_t + \alpha \frac{1}{M\ N} \sum_{m=1}^{M} \sum_{n=1}^{N} F_{mn} \left[\Delta W_m \circ \hat{} r_{mn} s_{mn} \right] \tag{2.24}$$

Where, N are the flipout perturbations at each weight, M the number of weights and $_F mn$ the reward at the $n - th$ perturbation at the weight m.

At this point, the loss function of the model is well defined by section 2.4, and the way to apply weight sampled from a distribution to a neural network via flipout method is illustrated in subsection 2.4.1, now, the way to train this networks is shown below in subsection 2.4.2:

2.4.2. Bayes by Backprop

Bayes by Backprop is an algorithm that trains Bayesian neural networks with uncertainty in their weights (figure 2.7). It optimizes a well defined loss function to learn the distribution on the weights of a neural network. To optimize the loss function it is necessary transform the expression in 2.20 into a computationally feasible one.

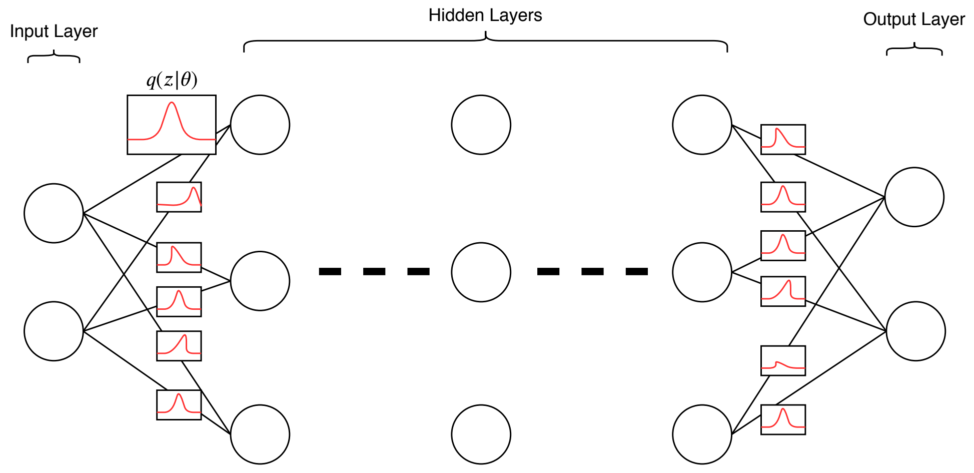


Figure 2.7: Scheme of a Neural Network with uncertainty in their weights

To transform the ELBO, first it is necessary to use next proposition[5]:

Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $w = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density *of z, $q(z|\theta)$, is such that $q(\epsilon)d\epsilon = q(z|\theta)dz$. Then for a function f with derivatives in w:*

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(z|\theta)}[f(z,\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(z,\theta)}{\partial z}\frac{\partial z}{\partial\theta} + \frac{\partial f(z,\theta)}{\partial\theta}\right]$$

Proof:

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{q(z|\theta)}[f(z,\theta)] &= \frac{\partial}{\partial\theta}\int f(z,\theta)q(z,\theta)dz\\ &= \frac{\partial}{\partial\theta}\int f(z,\theta)q(\epsilon)d\epsilon\\ &= \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(z,\theta)}{\partial z}\frac{\partial z}{\partial\theta} + \frac{\partial f(z,\theta)}{\partial\theta}\right] \quad \square\end{aligned}$$

This previous propositions allows to define a function $t(\theta,\epsilon)$, a deterministic distribution that can transform a sample of ϵ and the variational posterior parameters θ into a sample from the variational posterior, applying this to equation 2.20, it results in:

$$ELBO = log\ q(z|\theta) - log\ p(z) - log\ p(x|z) \tag{2.25}$$

Using Monte Carlo sampling to evaluate the expectations, similar as backpropagation algorithm, the Bayes by Backprop method is defined for variational Bayesian inference in neural networks which uses 2.20 to learn a distribution over the weights of a neural network. Finally, the complete aproximation of ELBO can be written as follows:

$$ELBO \approx \sum_{i=1}^{n} log\ q(z^{(i)}|\theta) - log\ p(z^{(i)})\ - log\ p(x|z^{(i)}) \tag{2.26}$$

where $z^{(i)}$ is the i_{th} Monte Carlo sample extracted from the posterior function $q(z^{(i)}|\theta)$. It is also important to note that each term in equation 2.26 depends on the extracted weights, in this case, a variance reduction technique known as common random numbers can be used [31].

Summarizing, Bayes by Backprop is an algorithm obtained to train Bayesian inference neural networks which uses unbiased estimates of gradients of the cost in 2.20 to learn a distribution over the weights of a neural network[5], the result of this algorithm is the equation 2.26 as unbiased loss function.

2.5. Putting all Pieces Together

The background presented above is necessary to understand the final purpose of this work, which is to use the temporality of the data to predict the behavior of some mechanical equipment and also be able to quantify the uncertainty in the RUL and health state prediction.

Quantifying uncertainty in this case refers to finding the probability distribution that most closely resembles the actual probability distribution presented by the data in relation to the variable to be predicted. To find this distribution, the **variational inference** method is presented, which transforms the problem of finding a probability distribution into an optimization problem.

Various techniques can be used to predict a probability distribution using artificial neural networks. In this work, the chosen technique corresponds to **Flipout**, which consists of treating the weights of a neural network as probability distributions centered on zero. With this, when entering the same value into the network multiple times, it will deliver different results that will allow the output distribution to be generated, that is, the distributions for the values entered into the network are obtained using the Monte Carlo method.

To make feasible the solution of the optimization problem that arises when using the variational inference to obtain the loss function and the flipout method to quantify the uncertainty, the cost function proposed in the section 2.4 must be adapted for a distribution obtained by the Monte Carlo method. The algorithm that approximates the cost function for this purpose is known as **Bayes by Backprop**.

Finally, the proposed models in this work combine the concepts explained above with the recurrent neural networks, that is, use the temporality of the data to predict a certain value and quantify the uncertainty in RUL and health state prediction.

Chapter 3

Methodology

To test the capabilities of Bayesian Recurrent Neural Networks in the context of PHM and thus, meet the objectives of this thesis, the methodological steps followed are shown below:

1. **Data Acquisition**: The models designed in this work are trained using a specific algorithm known as Bayes by Backprop, the recurrent networks are used to find specific representation of the data using the temporallity of it, in this case, the phenomenon of interest is the degradation of an asset or system that causes failures in it's operation. Therefore, the first step of this thesis is the acquisition of operational datasets of mechanical elements or systems, with the aim of validate the proposed models, in this case the chosen dataset is:

   - **CMAPSS Simulated Dataset**: The objective of this dataset is to perform prognosis of remaining useful life of turbofans using operational data generated by a simulation algorithm [36], this dataset is known for being one of the best dataset candidates for validate models in the context of RUL prediction.

   With the aim of testing the developed models, four more datasets were acquired:

   - **Fatigue Crack Growth University of Maryland**: This dataset is composed by the measurements of three disipative energies in a strain-stress essay developed in University of Maryland, the main objective for this data is to predict the RUL of each test sample with this data.
   - **Wind Turbine High Speed Bearings**: This data was collected by a condition monitoring system placed in a wind turbine, the data consists on the vibration of a bearing inside the turbine[2]. The main goal is to estimate the RUL of the bearings using this dataset to prevent a failure.
   - **University of Ottawa Bearings**: The data contains vibrational signals collected from bearings under time-varying rotational speed conditions, the experiments are performed on a SpectraQuest machinery fault simulator in University of Ottawa [18]. The aim of the study made with this dataset is to categorize the health state of the bearings.
   - **Politecnico di Torino Bearnigs**: This dataset was collected over a rig set up at DIRG Lab in the Department of Mechanical and Aerospace Engineering at Politecnico di Torino, the dataset consists on the measurement of acceleration,

rotational speed, radial load and damage level on bearings [8]. The objective is to test the ability of the developed models to use the temporality of this data and determinate the health state of it, this means that this is a classification problem.

2. **Literature Review:** Before the development phase of this work it is necessary to find the appropriate literature on both Bayesian Neural Networks and Deep Learning based methods applied to Reliability.
   In Bayesian Neural Networks the main topics necessary for the development of this thesis are shown below:

   - Variational inference to transform the problem from creating a distribution, to an optimization problem.
   - Weight perturbation to apply the *perturbated weights* to the studied models.
   - Bayes by Backprop to transform the optimization problem into a computationally feasible problem.

3. **Development of B-RNN:** This step consists in both the theoretical adaptation of *weight perturbation* to work in Recurrent Neural Networks, and the development of such adaptation in code.

   The variational inference tool is used to develop the probabilistic model necessary to quantify uncertainty, that is, the problem of finding a probability distribution becomes an optimization problem, this concept serves as a theoretical basis for the development of networks recurrent Bayesian neural neuroses developed in this work.

   In this work the probabilistic model developed is based on the **flipout method** explained in section 2.4.1, this method is implemented in dense layers in the *tensorflow probability* library, in this thesis these layers are used to replace the logic gates inside the recurrent cells JANET and VRNN, in this way the weights within the developed recurrent layers are treated as probability distributions and are capable of propagating uncertainty, finally, as in frequentist recurrent neural networks, the weights of these networks are shared to the different time steps within each of the data delivered to the network, this, at the code level is done using a "for"loop for each time step.

   Finally, to develop the training of the developed networks, the Bayes by Backprop algorithm is implemented, which, as explained above, is an algorithm developed to train neural networks whose weights are treated as probability distributions, to carry out this implementation the tools are used from the *tensorflow probabilities* library to calculate the KL divergence and the term corresponding to the log likelihood in each epoch, after this, the keras optimizer **Adam** is used to update the parameters of the distributions from which the network weights are extracted.

4. **Test and Validation in Simulated Dataset:** Once the model is fully developed, the next step is testing it in simulated data, this is to be sure that the model can get results from it.
   The use of turbo fan data is also a good way to compare the performance of the proposed model with other models previously designed for the prediction of RUL in this dataset.

5. **Test in Real Dataset:** After testing the model in the validation data, it is important to show the performance of the model in real data, for this, the four acquired data are tested using the developed models.

6. **Analysis of the results and concluding remarks:** After the experiments are finished, the results can be analyzed and the hypotheses will be either refuted or validated Also, from this analysis, the concluding remarks of this thesis will be drawn.

Chapter 4

Proposed Bayesian Recurrent Neural Networks

This chapter presents the recurrent Bayesian neural networks developed and implemented in this work.

The development of recurrent Bayesian neural networks consists of the development of recurring layers based on flipout layers and the implementation of the Bayes by Backprop algorithm to train the networks.

4.1. Bayesian Recurrent Neural Networks

The proposed Bayesian Recurrent Neural Networks (B-RNNs) are recurrent neural networks in which weights are extracted from a probability distribution using flipout method and that are also able to deliver a probability distribution as output. Because of this, in the case of RUL predictions one B-RNNs model is able to get multiple samples and get the probability distribution for every point in the dataset. In the case of diagnosis, the distribution is discrete, so the network will tend to choose the correct class.

As it is possible to get the probability distribution for each point in the dataset in the case of prognosis, it is also possible to get the probability intervals from this distribution. This concept is interesting in PHM context because the confidence interval in RUL represents the probability of failure of a determinate mechanical equipment in a specific time with a certain confidence.

In the following Figure, there is a flowchart of the developed B-RNN approach to make it more understandable.

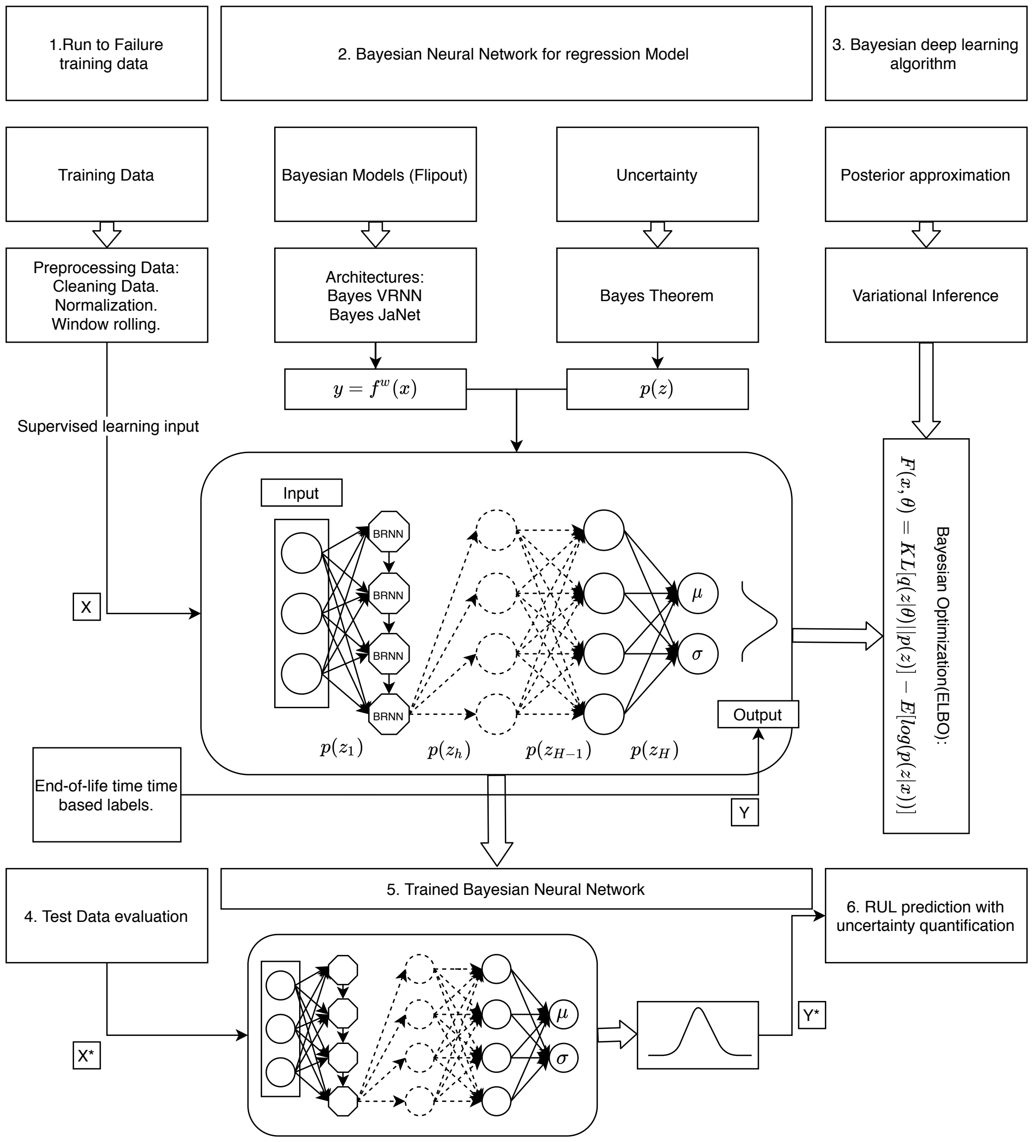

1.Run to Failure training data
2. Bayesian Neural Network for regression Model
3. Bayesian deep learning algorithm
Training Data
Bayesian Models (Flipout)
Uncertainty
Posterior approximation
Preprocessing Data: Cleaning Data. Normalization. Window rolling.
Architectures: Bayes VRNN Bayes JaNet
Bayes Theorem
Variational Inference
$y = f^w(x)$
$p(z)$
Supervised learning input
Input
BRNN
X
μ
σ
Output
$p(z_1)$
$p(z_h)$
$p(z_{H-1})$
$p(z_H)$
Bayesian Optimization(ELBO):
$F(x,\theta) = KL[q(z|\theta)||p(z)] - E[log(p(z|x))]$
End-of-life time time based labels.
Y
4. Test Data evaluation
5. Trained Bayesian Neural Network
6. RUL prediction with uncertainty quantification
X*
Y*

Figure 4.1: Framework of the proposed Bayesian RNN's models.

As shown in chapter 2, RNNs make a prediction one step ahead with dependency on the present state and all previous time-step outputs (hidden states). This means that a k-step prediction depends on the hidden states for $t \in 0, ..., k-1$ and timestep k and so on. The developed Bayesian RNN's of this work performs in the same way, but the main difference is the way that the weights of the recurrent cell operates. In the case of the developed B-RNN's the weights are treated as distributions and are applied using flipout method illustrated in section 2.4, applying the equation of flipout 2.4.1 [44] to the basic equation of RNN 2.4. The next equation is developed for Bayesian Vanilla RNN (B-VRNN):

$$f_t = \sigma(\bar{U_f}^T h_{t-1} + (\Delta\hat{U_f}^T (h_{t-1} \circ s_t^h)) \circ r_t^h + \bar{W_f}^T x_t + (\Delta\hat{W_f}^T (x_t \circ s_t^x)) \circ r_t^x)$$
$$h_t = f_t \tag{4.1}$$

Where $t \in 0, ..., k$ is the discrete-time index with k the prediction horizon, h_t is the hidden state of the cell representing time-step t, x_t the cell input, $\bar{W_i}$, $\bar{U_i}$ are the means of the weight matrix and $\Delta\hat{W_i}$, $\Delta\hat{U_i}$ are the "perturbation" of the weight matrix that is applied to the input and hidden vector respectively.

In conclusion, $\bar{W_i}$, $\bar{U_i}$, $\Delta\hat{W_i}$, $\Delta\hat{U_i}$ are trainable parameters of the developed Bayesian recurrent layers, these parameters are learned from training data by minimizing the sum between the negative log likelihood and the Kullback-Leibler using Bayes by Backpropagation algorithm.

The developed equations for the proposed B-RNN's in all the studied Bayesian cells are presented below, these equations are generated by applying the flipout estimator to the frequentist RNN cells.

- The operation inside Bayesian VRNN are:

$$\begin{gathered} f_t = \tanh\Big(\bar{U_f}^T h_{t-1} + (\Delta\hat{U_f}^T (h_{t-1} \circ s_t^h)) \circ r_t^h + \bar{W_f}^T x_t + (\Delta\hat{W_f}^T (x_t \circ s_t^x)) \circ r_t^x\Big) \\ h_t = f_t \end{gathered} \tag{4.2}$$

- Bayes JaNet Operations:

$$\begin{gathered} f_t = \sigma(\bar{W_f}^T x_t + (\Delta\hat{W_f}^T (x_t \circ s_t^x)) \circ r_t^x + \bar{U_f}^T h_{t-1} + (\Delta\hat{U_f}^T (h_{t-1} \circ s_t^h)) \circ r_t^h) \\ c_t = f_t \circ h_{t-1} + (1 - f_t) \circ tanh(\bar{U_c}^T h_{t-1} + (\Delta\hat{U_c}^T (h_{t-1} \circ s_t^h)) \circ r_t^h + \\ \bar{W_c}^T x_t + (\Delta\hat{W_c}^T (x_t \circ s_t^x)) \circ r_t^x) \\ h_t = c_t \end{gathered} \tag{4.3}$$

All the above equations follows the same notation, where:

- $\bar{W_i}$, $\bar{U_i}$: Represent the mean of the weight matrix that is applied to the input and hidden vector respectively.
- $\Delta\hat{W_i}$, $\Delta\hat{U_i}$: Represent the "perturbation" of the weight matrix that is applied to the input and hidden vector respectively.
- R, S: Are random matrices whose entries are sampled uniformly from ± 1.

It is important to note that the schemes of the cells in B-RNN's are the same that the frequentist networks.

4.2. Training of Bayesian Recurrent Neural Networks

The main objective of this Bayesian model in mathematical terms, is to minimize the negative ELBO as it is explained in section 2.4, for this, B-RNN's has to minimize the equation 2.26. The process of trainig B-RNN's is similar to the process of regular recurrent neural networks, this means that the process of optimization is done by computing the gradient of each parameter of θ (corresponding to the parameters of the distributions from where the weights of the network are sampled) with respect to the loss function L (equation 2.26), this is $\frac{\partial L}{\partial \theta}$. This derivative is calculated via Bayes by Backpropagation algorithm, which means that it is necessary to sample from the distribution of the weights ($q(z|\theta)$) to calculate the derivative illustrated above. Finally, gradient descent algorithm is used to update each parameter.

Bayesian Recurrent Networks (B-RNN) are studied in [44], where Bayes by Backprop is validated in recurrent networks. The approach to generate B-RNNs is to wrap the Tensorflow probability layers. With this, it is possible to get distributions as outputs in the developed models, encouraging the time prediction of the RNNs and including uncertainty about the results. This opens many possibilities in reliability engineering where decisions need to be made considering risk.

4.3. Performance Metrics for Bayesian Recurrent Neural Networks

Since the output of the proposed model is a distribution, it is not possible to use the metrics described in subsection 2.3.4. Therefore, it is necessary to adapt the model output to a value comparable to the commonly used models in terms of accuracy and mean square error.

To obtain values comparable to frequentist models, it is necessary to obtain samples of the model's output distributions, and use the average of these values as a prediction point, to finally use this value to calculate the metrics corresponding to the regression and classification tasks.

For the regression, the equations to calculate the RMSE is described below:

$$RMSE = \sqrt{\sum_{i=1}^{N}(y_i - \hat{\hat{y}}_i)^2} \tag{4.4}$$

where y_i is the real value of point x_i and $\hat{\hat{y}}_i$ is the mean of the distribution generated from the data x_i.

4.4. Implementation

For the implementation of the proposed model, advantage is taken of the fact that each of the gates within a recurring cell consists of a layer of an artificial neural network. Therefore, to add uncertainty in the weights of the recurring network, the gates within the recurring cells are replaced by DenseFlipout layers developed in the library of Tensorflow Probability [12].

Each of the weights of the proposed Bayesian recurrent neural networks are represented by a normal distribution, therefore, each of these weights has two parameters, a mean μ and a standard deviation σ.

A problem that arises when training the recurrent Bayesian neural network developed in this work consists in the difference in the orders of magnitude of the two terms in the loss function presented in equation 2.26. To solve this, a regulator must be chosen that allows both terms in the loss function to have a similar order of magnitude, the choice of this parameter is different for each dataset and is shown in the section corresponding to each one.

Chapter 5

Case Studies

5.1. Validation Dataset: RUL Prediction in Turbofan Engine Degradation Simulation (CMAPSS)

The first case study presented in this thesis is a prognosis example corresponding to a simulated data. The aim of this dataset is the estimation of remaining useful life in turbofan engines. The data for this case study comes from CMAPSS originated for the PHM 08' data competition [36] and used since then as a benchmark dataset for PHM purposes. A scheme of the Simulated turbofans is shown in Figure 5.1. This chapter starts with description of the dataset followed by the preproccesing necessary to use the proposal approach of section 4. Finally, the results obtained are presented below.

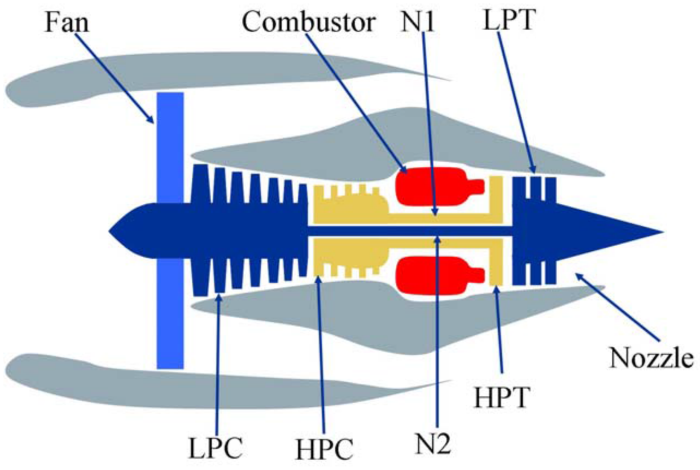


Figure 5.1: Scheme of The turbofans simulated by NASA

5.1.1. Data Description

Dataset is composed of multivariate operational time-series obtained by simulating a turbofan under various fault modes and operational conditions using the *Commercial Aero Propulsion System Simulation* (CMAPSS) [36]. This dataset is organized in four categories depending on the operational condition and fault modes of the turbofans, this organization of the dataset is shown in the table below.

Table 5.1: Caracterization of CMAPSS

Subset	Operational Conditions	Fault Modes
FD001	One	One (HPC Degradation)
FD002	Six	One (HPC Degradation)
FD003	One	Two (HPC and Fan Degradation)
FD004	Six	Two (HPC and Fan Degradation)

The description of the sensors and their measurement units is illustrated in table below.

Table 5.2: C-MAPSS sensors

Sensor	Descriprion	Unit
#1	Total temperature at fan inlet	°R
#2	Total temperature at LPC outlet	°R
#3	Total temperature at HPC outlet	°R
#4	Total temperature at LPT outlet	°R
#5	Pressure at fan inlet	psia
#6	Total pressure in bypass-duct	psia
#7	Total pressure at HPC outlet	psia
#8	Physical fan speed	rpm
#9	Physical core speed	rpm
#10	Engine pressure ratio (P50/P2)	-
#11	Static pressure at HPC outlet	psia
#12	Ratio of fuel fow to Ps30	pps/psi
#13	Corrected fan speed	rpm
#14	Corrected core speed	rpm
#15	Bypass Ratio	-
#16	Burner fuel-air ratio	-
#17	Bleed Enthalpy	-
#18	Demanded fan speed	rpm
#19	Demanded corrected fan speed	rpm
#20	HPT coolant bleed	lbm/s
#21	LPT coolant bleed	lbm/s

In the dataset, each row represents one time-step measured in cycles and each column is a different sensor in the order showed in table 5.2. As this dataset was created for a competition, the train-test sets were already separated, and in both cases the trajectories are simulated from a normal operational state with an unknown level of degradation, but the difference between the train and test set is that the train trajectories are simulated until the engine presents one of the corresponding fault modes and the test set simulated trajectories are stopped at a random moment before that. Nowadays, the RUL for the test examples have been made available to the public with the rest of the data.

The following section explains the preprocess of the data to be tested using the proposal methods.

5.1.2. Data Preparation

The first step for the preprocessing of CMAPSS dataset is to eliminate the sensors that don't change in time, this allows to erase unnecessary information. For datasets FD001, FD002, FD003 and FD004 the number of sensors eliminated is 7, which means that the total useful sensors remaining for the following preprocessing is 14.

As this dataset is used on a competition, the train and test were already separated, knowing this, a scaler is trained with the training data to normalize the testing data, this is a common process used in machine learning to accelerate and stabilize the training process of the models.

The final step is to increase the number of training data-points and standardize the length of them, since the proposed approach is not designed to support inputs of variable length. One common strategy for this is to generate moving windows of length T time-steps, which slides over each trajectory with an overlap v. This process is illustrated in figure 5.2 .

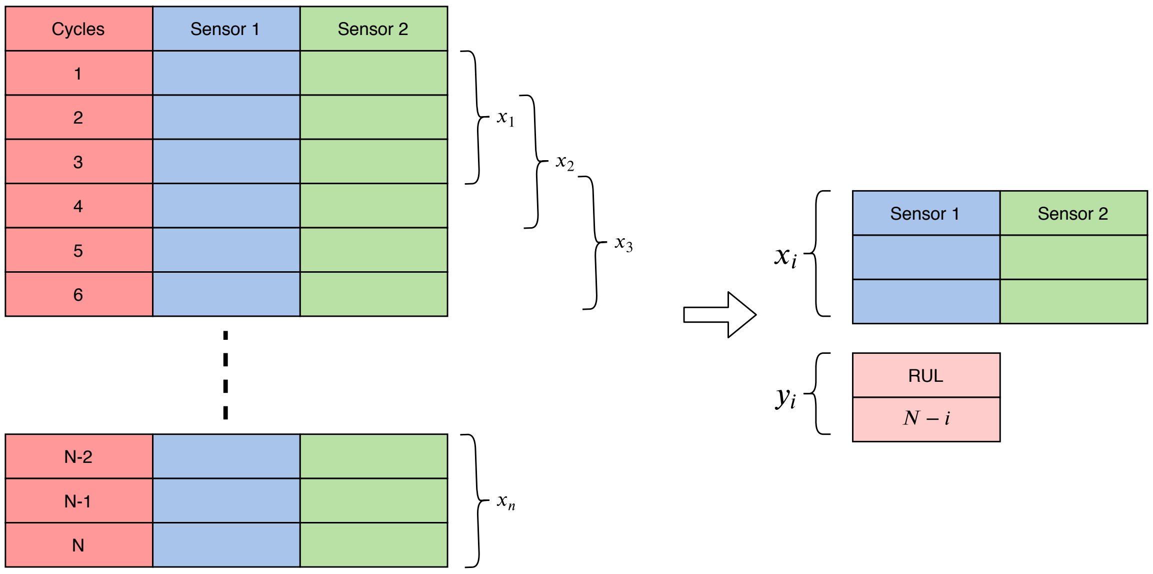


Figure 5.2: Example for preprocessing of CMAPSS Dataset, in the example the overlaping is 2 and the timestep T is 3.

Finally, every data point consists in the measurement of 14 sensors and variable time-steps according to the study set. The overlap in every data is one step less than the time-step T. Table 5.3 shows the final structure of data points according to their respective dataset.

Table 5.3: Data shape for different datasets on CMAPSS

Dataset	Time Step	Sensors
FD001	30	14
FD002	21	14
FD003	30	14
FD004	18	14

In the following section, the four datasets are tested in both proposal architectures of this thesis, Bayes Vanilla RNN and Bayes JaNet.

5.1.3. Results for Bayes VRNN Cell in CMAPSS Dataset

In this section, the results for Vanilla Recurrent neural network are illustrated for each one of the four datasets in table 5.3, the optimization of the parameters is done via grid search, and the results for the parameters are illustrated in table below.

Table 5.4: Parameters for Bayesian VRNN in CMAPSS dataset.

Layer	Units/Neurons	Activation Function
BayesVRNN	32	
DenseFlipout	32	Relu
DenseFlipout	16	Relu
DenseFlipout	2	

The final Bayesian layer has two neurons, this is because the output of the network is a normal distribution that has two parameters, the mean(μ) and the standard deviation(σ). The parameters in this networks are twice than their frequentist version, this is because every weight in the developed Bayesian layers has two parameters instead of one.

The number of epochs used for this model is 100, this is because at this point the loss function 2.26 is stable and minimized. The regulating factor used for the stabilization of the KL divergence is $1e-6$ and the samples took from the model to generate the distribution of the test samples is 100.

The Bayesian recurrent layer returns all the sequence of data and not just the last value, a scheme of the network used is shown in figure 5.3.

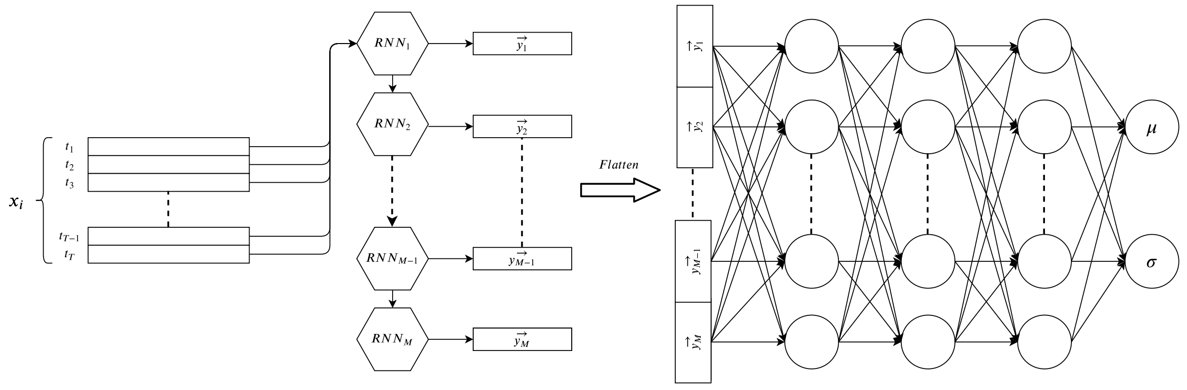
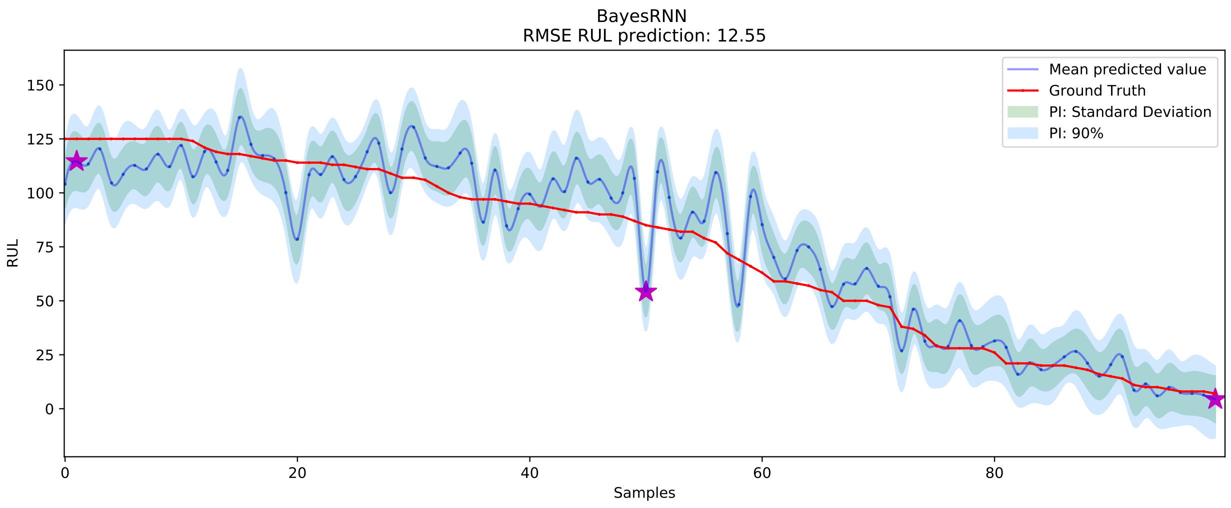


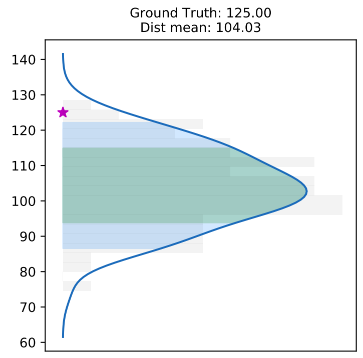
Figure 5.3: Scheme of RNN model used.

Where M is the number of units in RNN and T is the time-step defined for the dataset in section 5.1.2. As it was shown in section 2 RNN's take a temporal series as input and the output could be a single vector obtained from the last recurrent cell or a vector for each unit in the network, in this case every vector of every recurrent cell is used. To pass this vector to the *DenseFlipout* layers, a Flatten operation is applied.

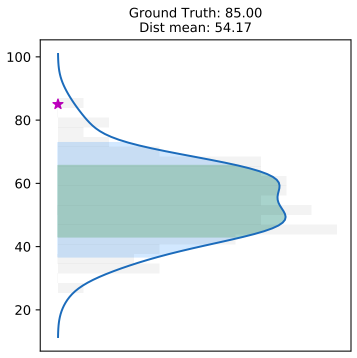
- FD001:
  Below the results for Bayesian VRNN are presented for RUL prediction in FD001 dataset. This is the easiest dataset among CMAPSS simulation, because of the operational conditions and the fault modes.



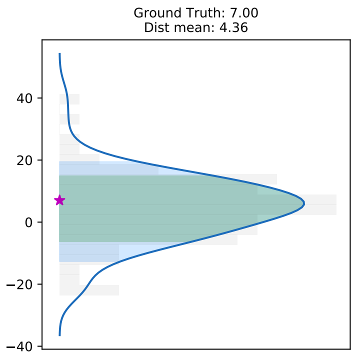
(a) Comparison between real and predicted RUL in CMAPSS FD001 data



(b)



(c)



(d)

Figure 5.4: RUL Prediction for VRNN architecture on FD001 CMAPSS dataset, The (a) image is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 50 and 100 respectively.

As it is shown in figure 5.4 the uncertainty is quantified for every point in the test set, it is important to observe that almost every real point in the dataset is inside of the distribution predicted by the model. The probability distribution for each of the points in the test set is different as it's shown in figures 5.4b, 5.4c and 5.4d, this is because each point in the test set has its own features and properties that can make more or less difficult the prediction task.

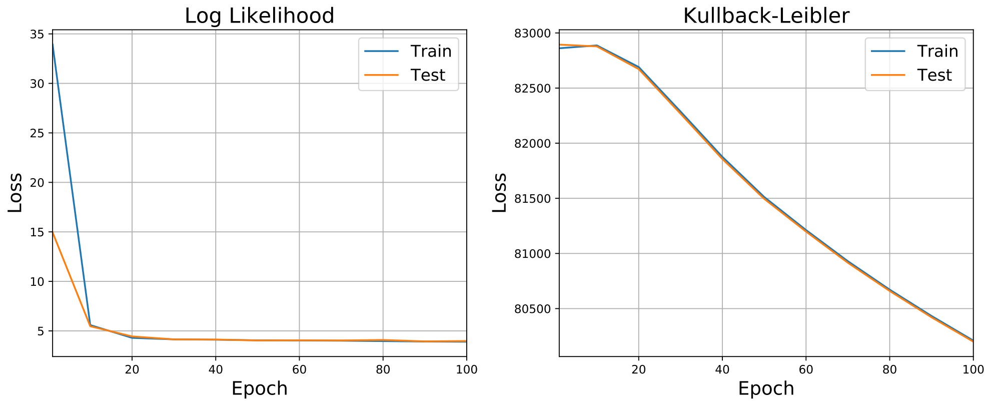
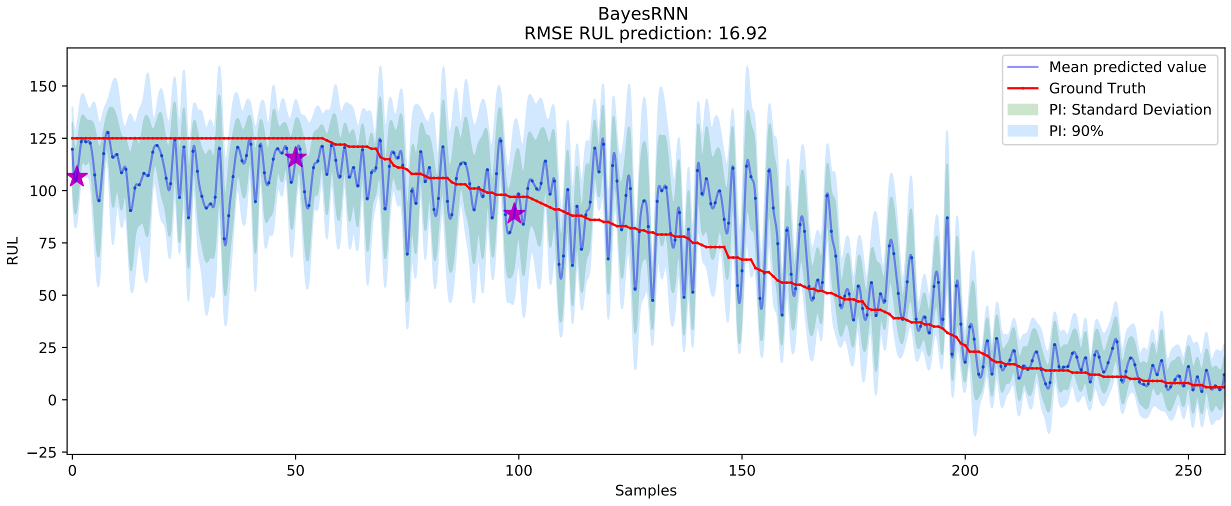


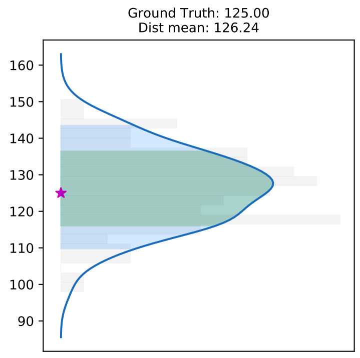
Figure 5.5: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD001 dataset.

In figure 5.5 both, the KL divergence and the log likelihood are decreasing, this means that the difference between the real distribution function and the proposed function are reducing the *distance* between them, as each term has its own meaning, the log likelihood term measures how well the model fits to a sample of data for given values of the unknown parameters, so its decrease means that the predicted values are getting closer to the real ones. The decrease in KL divergence means that the proposed distribution is getting closer to the prior distribution as the parameters of the weight distributions are updated, it is important to note that this does not mean that the prediction will improve further if the KL divergence continues to decrease.

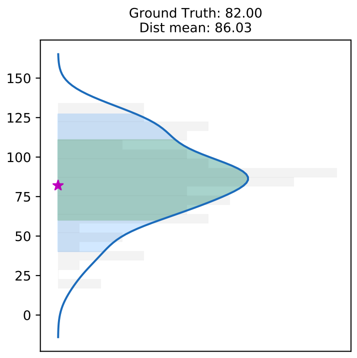
- FD002:
  This dataset is much more complex than FD001 due to the quantity of operational conditions, in the case of FD002 there are six operational conditions while in FD001 there is only one. The results for this dataset are shown below:



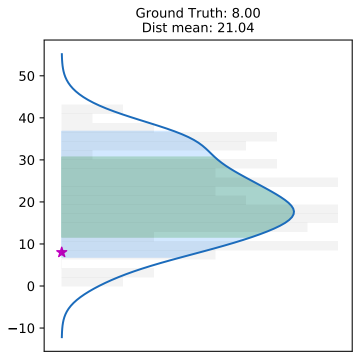
(a) Comparison between real and predicted RUL in CMAPSS FD002 data



(b)



(c)



(d)

Figure 5.6: RUL Prediction for VRNN architecture on FD002 CMAPSS dataset, The (a) image is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 125 and 250 respectively.

As it can be seen in figure 5.6 the results are not as good as in the first dataset, this is expected considering the complexity of the current data, however, an interesting remark is that the standard deviations of the distributions in figures 5.6b, 5.6c and 5.6d is bigger than in the first dataset due to the difference in the complexity of the current data.

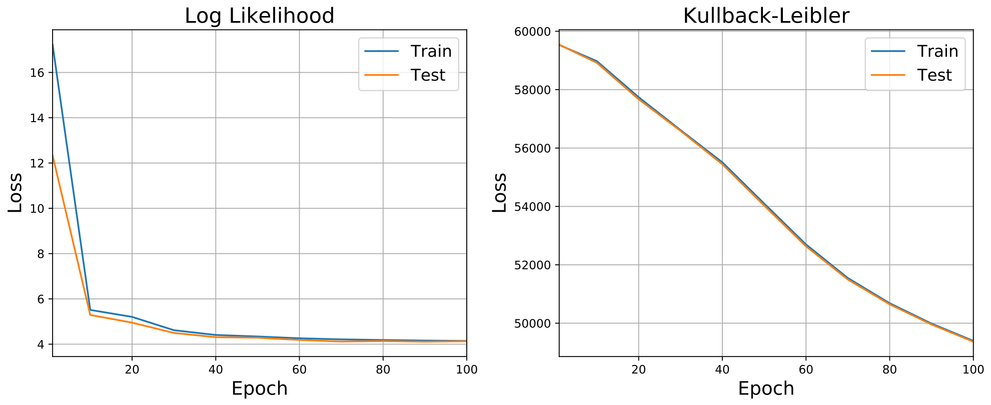
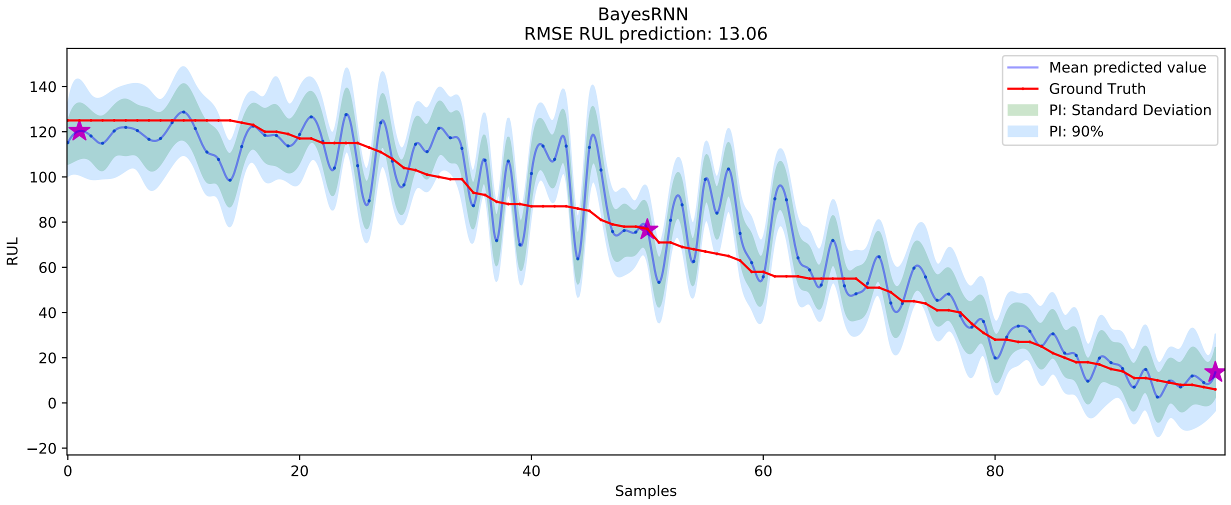


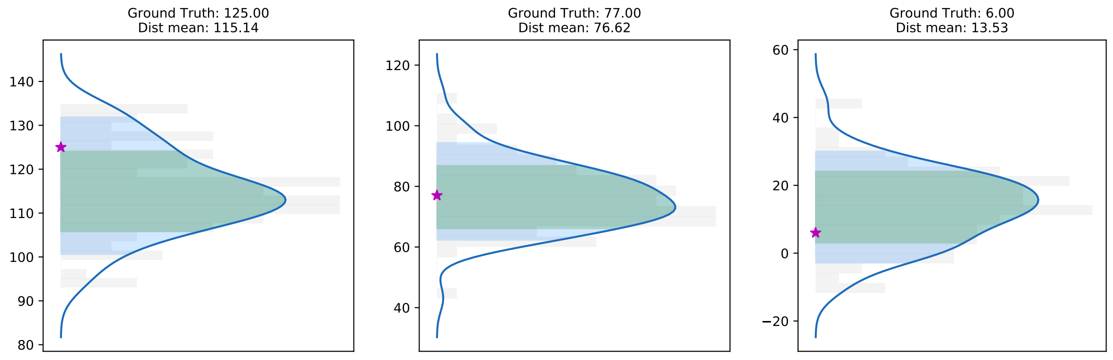
Figure 5.7: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD002 dataset.

The figure above shows the evolution of the terms in the loss function across the epochs of training, in this case the evolution and thus the conclusions about this behavior are the same that in the first dataset.

- FD003:
  This dataset consists in simulations of turbofans with one operational condition and two fault modes, this means that this dataset is more complex than FD001, but less complex than FD002, the result for this dataset is showed below in figure5.8.



(a) Comparison between real and predicted RUL in CMAPSS FD003 data



(b) (c) (d)

Figure 5.8: RUL Prediction for VRNN architecture on FD003 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 50 and 100 respectively.

As it is expected, the results are similar to FD001 dataset, but the standard deviation are slightly bigger, this is because the model is not able to predict the RUL with the same certainty that in the first dataset, however, the predictions are clearly accurate and the uncertainty is quantified.

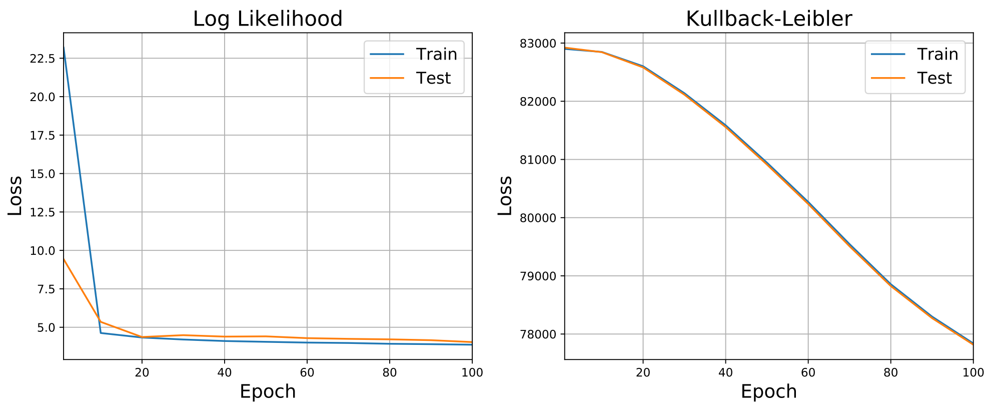
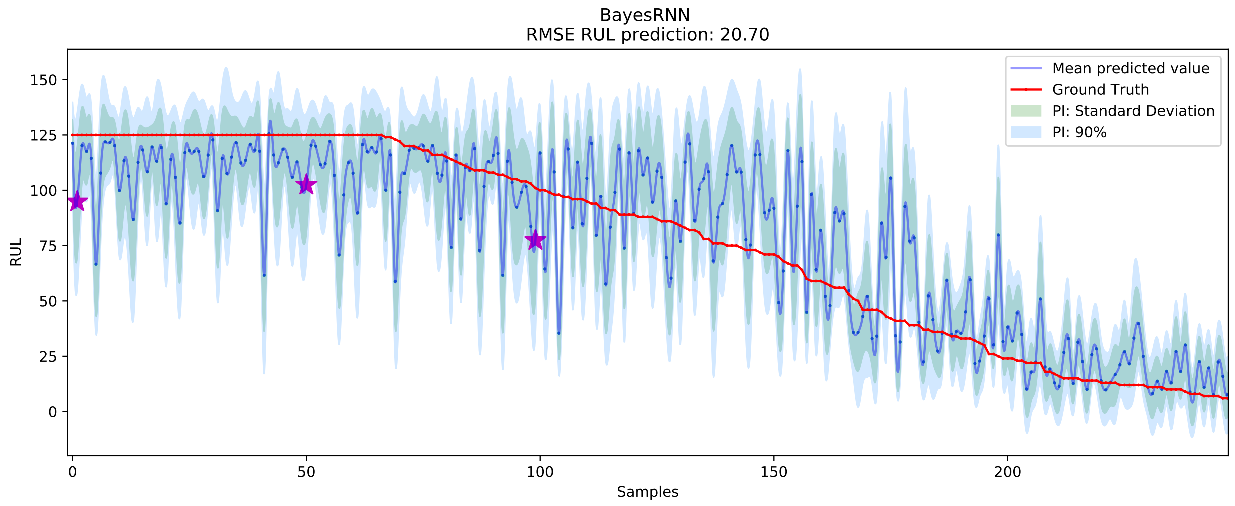


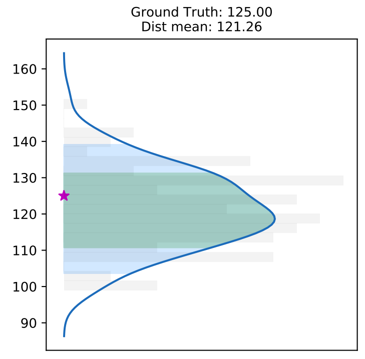
Figure 5.9: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD003 dataset.

The behavior showed in figure 5.9 is the same that in figure 5.5.

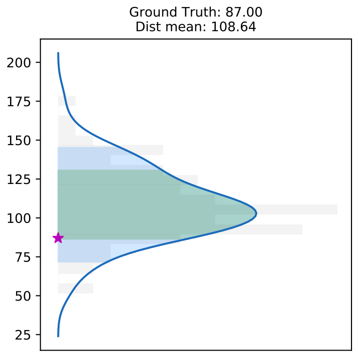
- FD004:
  This is the most complex of the four datasets due to the simulated operational conditions and the multiple fault modes. The results for this dataset are shown below in figure 5.10.



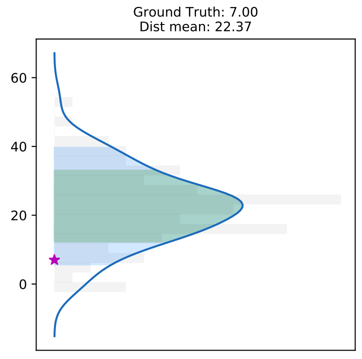
(a) Comparison between real and predicted RUL in CMAPSS FD004 data



(b)



(c)



(d)

Figure 5.10: RUL Prediction for VRNN architecture on FD004 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 125 and 245 respectively.

The results of this dataset are clearly worst than in previous datasets, which means that the final RMSE is worst and the standard deviation is bigger. However, an important observation is that in the majority of the test samples, the real RUL value belongs into the range of the predicted distribution but this estimation is not as valid as in the previous samples, this is because the standard deviation is large compared to the magnitude order of the RUL, therefore, since the dataset is more complex, the uncertainty of the model increases at the time of predicting the data, resulting in less-accurate predictions.

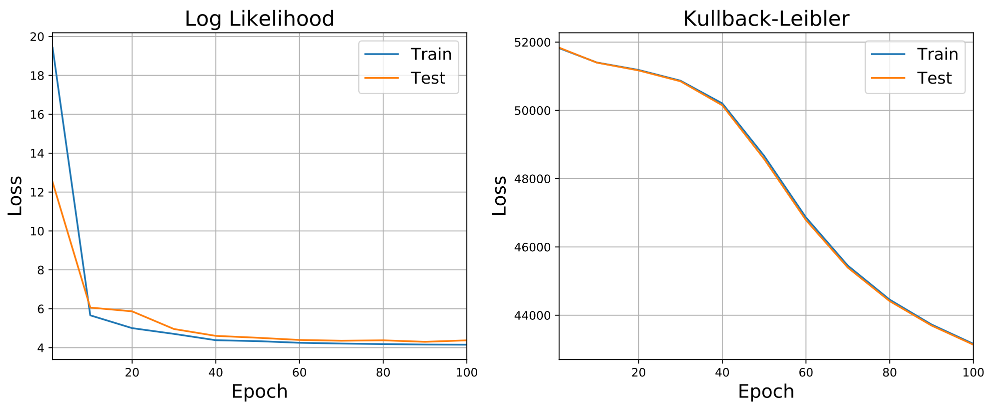


Figure 5.11: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD004 dataset.

The behavior showed in figure 5.11 is the same that in figure 5.5.

5.1.4. Results for Bayes JaNet Cell in CMAPSS Dataset

In this section, the results for Bayesian Just Another Networks (JaNet) in CMAPSS dataset are presented, the optimization of the hyperparameters is the same that in subsection 5.1.3 and the results for the networks parameters are illustrated in the table below:

Table 5.5: Parameters for Bayesian JaNet in CMAPSS dataset.

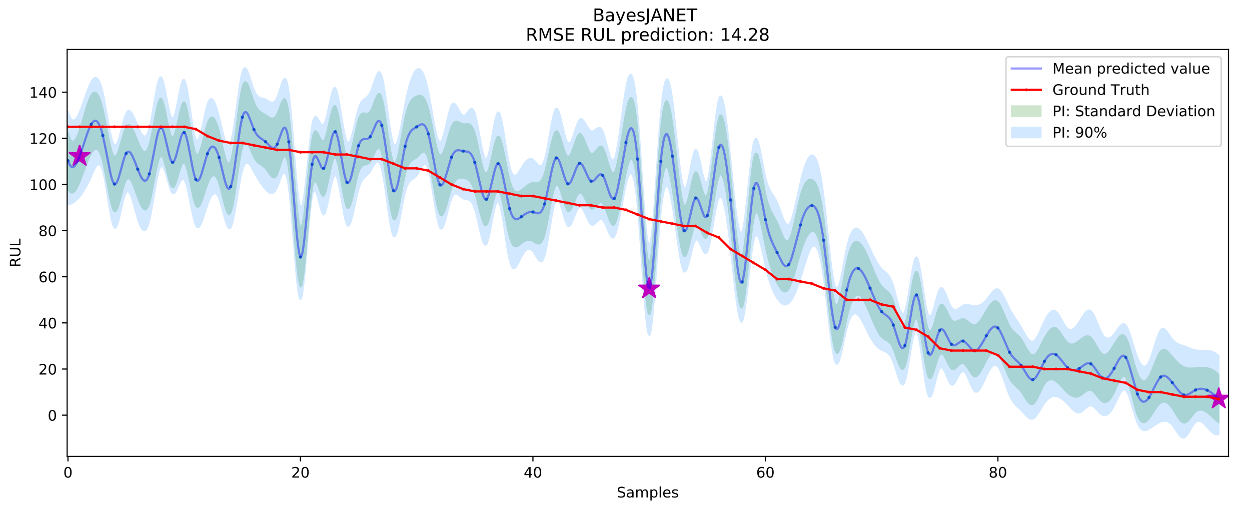
Layer	Units/Neurons	Activation Function
BayesJaNet	32	
DenseFlipout	32	Relu
DenseFlipout	16	Relu
DenseFlipout	2	

The number of epochs for this training process is fixed in 100 because at this point the loss function is stable, the regulating factor is $1e-6$ and the samples took from the model to generate the distribution of the test samples is 100.

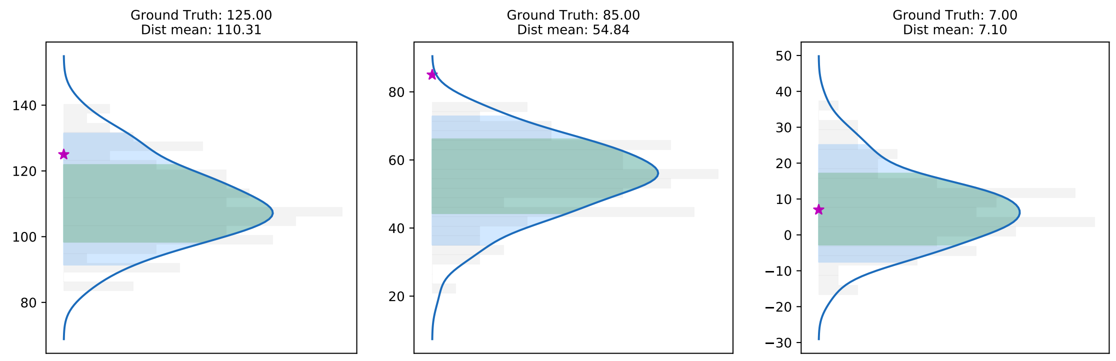
The network structure is the same that the network illustrated in figure 5.3, which means that for this process, all the sequential data obtained from the recurrent network is used to train the model.

Below are the results of the four CMAPSS datasets for the architecture described above.

- FD001:
  The results for CMAPSS FD001 are presented in figure 5.12. As it can be seen in this figure, the value of the global RMSE is greater than in the case of the B-VRNN and in the case of the standard deviations, the values do not demonstrate a significant change.



(a) Comparison between real and predicted RUL in CMAPSS FD002 data



(b) (c) (d)

Figure 5.12: RUL Prediction for JaNet architecture on FD001 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 50 and 100 respectively.

Although the results for this architecture are not better than for the VRNN, it is important to note that most of the real values are found within the predicted distributions.

The evolution of the log likelihood and the Kullback Leibler is presented in the figure below:

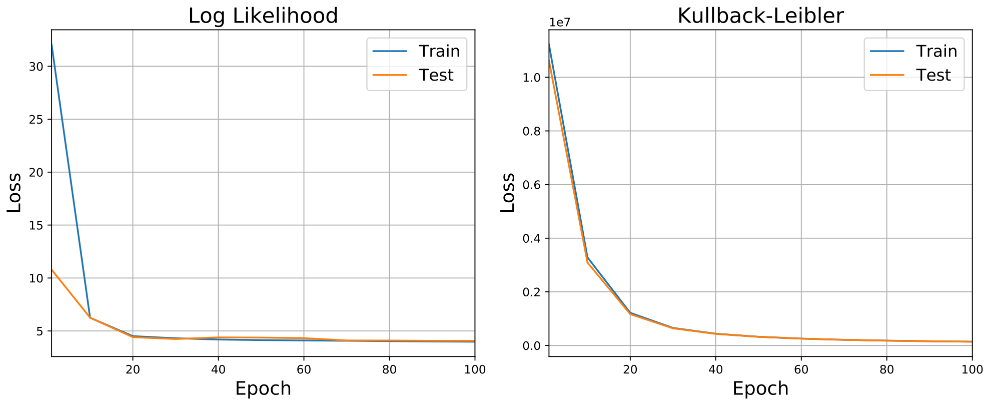
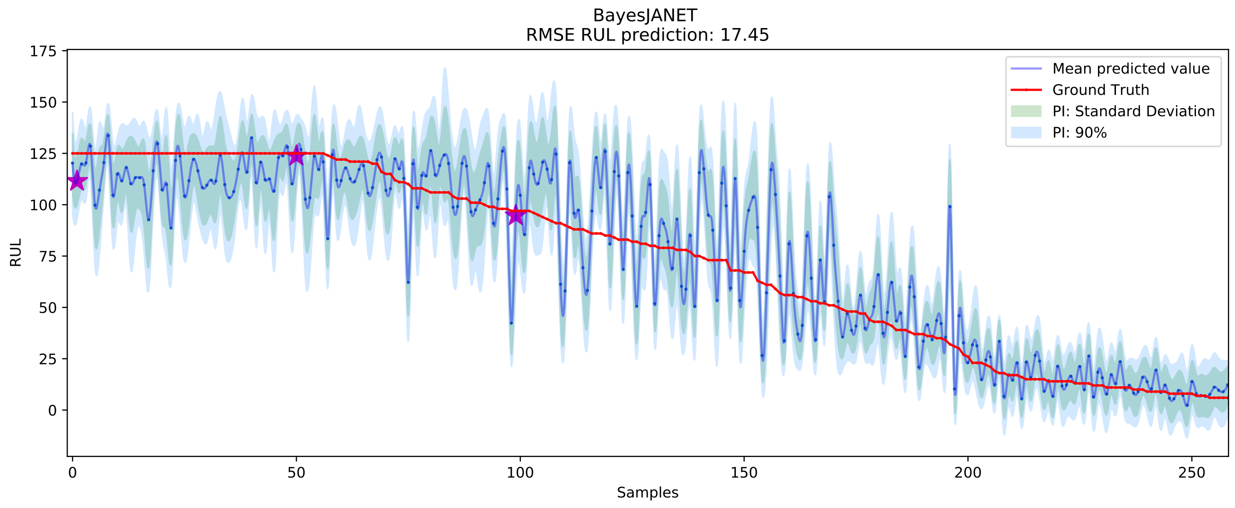


Figure 5.13: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD001 dataset.

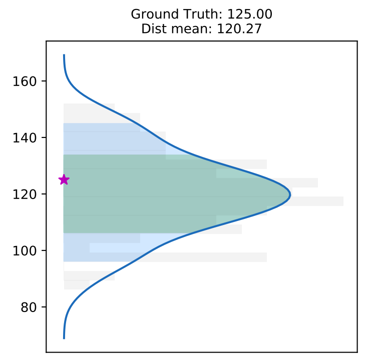
Both, the Kullback-Leibler and the log likelihood decrease, which mean that the proposed distribution is getting closer to prior one while the values predicted are getting similar to the real ones.

As it can be seen in figure 5.13, the orders of magnitude of the two components of the cost function used in this model are very different, this is because there is no restriction for the Kullback Leibler value scale, moreover, the greater the number of parameters, the greater the value of this term, which is why the regulator used is the inverse of the difference between the value scales.

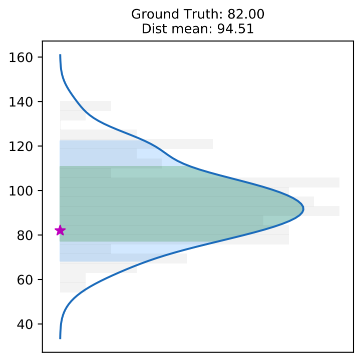
- FD002:
  This dataset, as it was explained before, is more complex than FD001, but as it seen in figure 5.14, the model can understand the behavior of the data, and predict in a reasonable way the RUL of it.



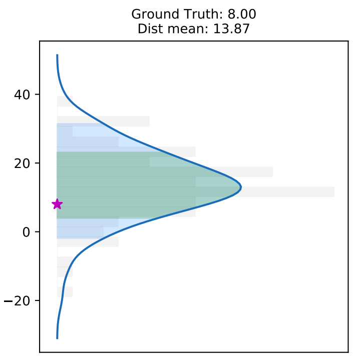
(a) Comparison between real and predicted RUL in CMAPSS FD002



(b)



(c)



(d)

Figure 5.14: RUL Prediction for JaNet architecture on FD002 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 125 and 250 respectively.

Taking into account the difficulty of the data, the results are acceptable from the point of view of determining uncertainty, despite the fact that the predictions are not optimal, in most cases, the model is capable of encompassing the actual value of the RUL within the probability distribution.

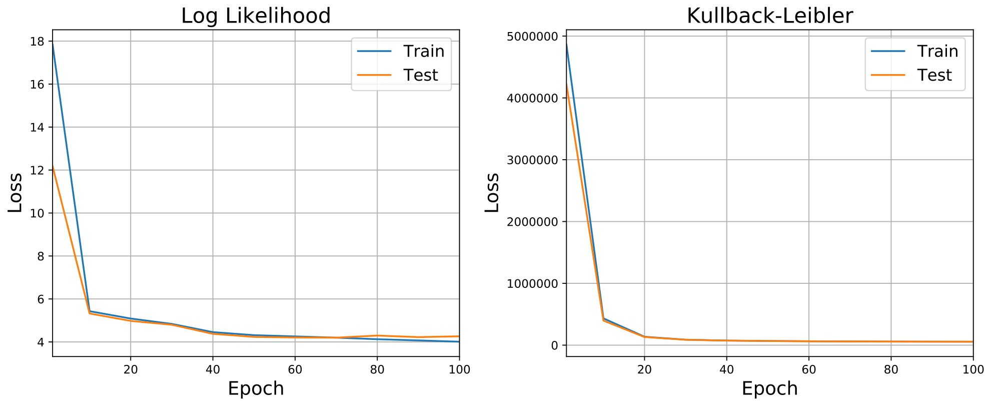
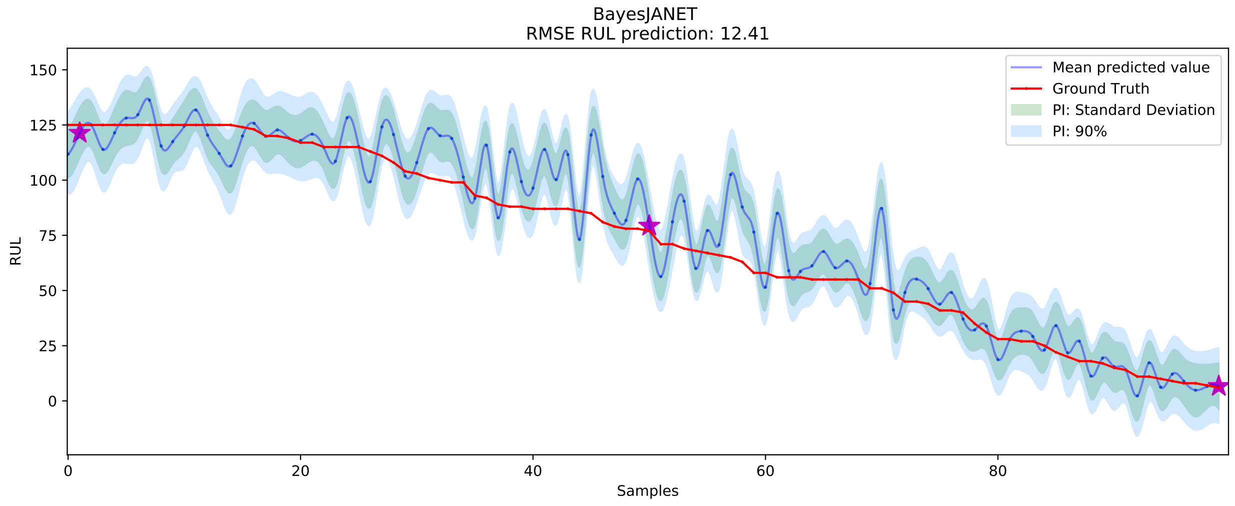


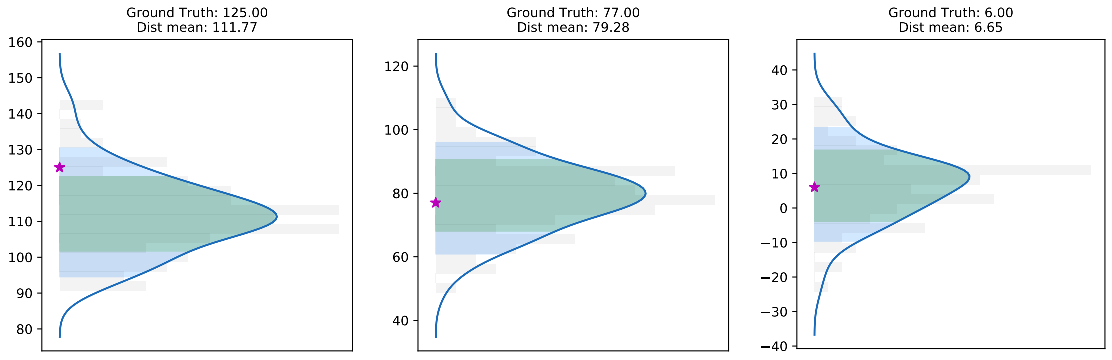
Figure 5.15: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD002 dataset.

The behavior of the loss function illustrated in this image is identical to the one illustrated in figure 5.13, therefore, the observations are similar.

- FD003:
  The results obtained for the FD003 dataset are presented below in figure 5.16, in which it is distinguished that the behavior of the data is correctly interpreted by the proposed model, in addition, in most cases, the resulting distributions of the model include the actual value of the RUL.



(a) Comparison between real and predicted RUL in CMAPSS FD003 data



(b) (c) (d)

Figure 5.16: RUL Prediction for JaNet architecture on FD003 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 50 and 100 respectively.

As it is possible to observe in figure 5.17, the loss function has the same behavior as the one illustrated in figure 5.13, which means that the model is trained and regulated in a stable way.

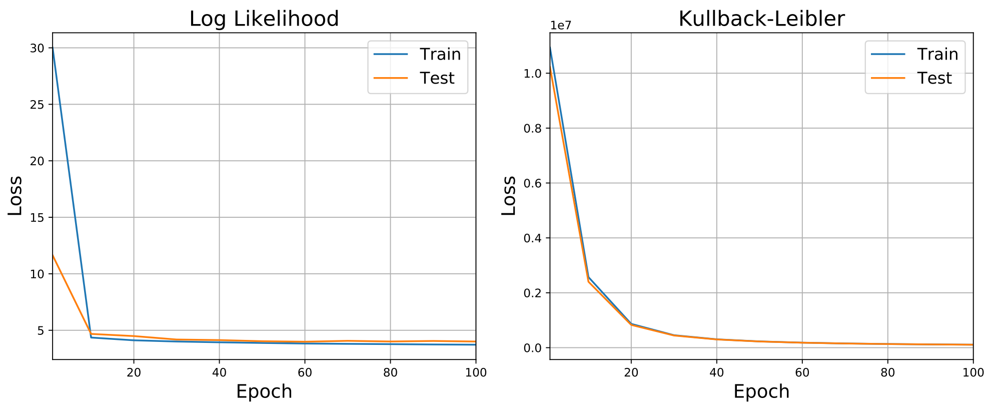
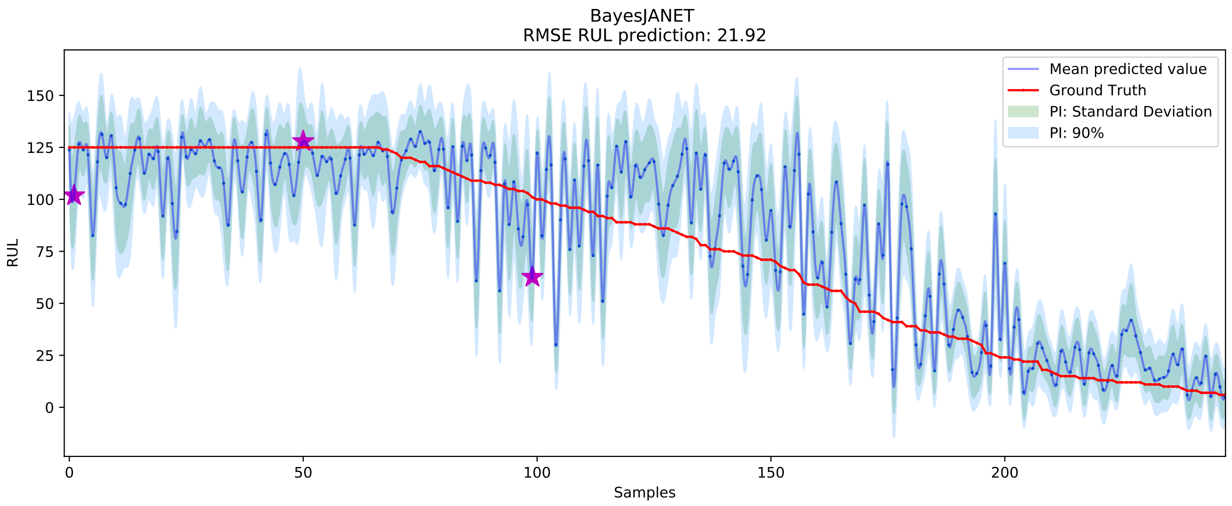
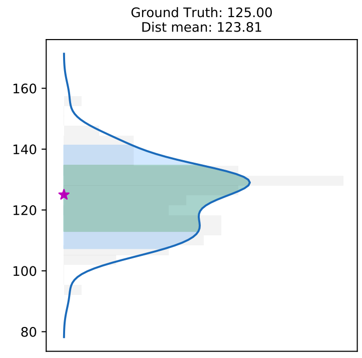


Figure 5.17: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD003 dataset.

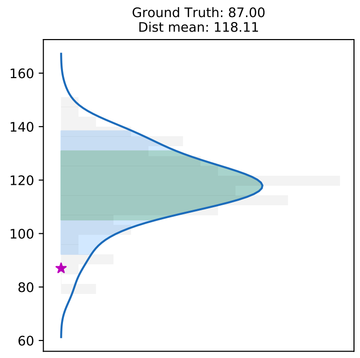
- FD004:
  This dataset is the most complex among the four, the results obtained for the FD004 dataset using the JANET cells are illustrated below in figure 5.18.



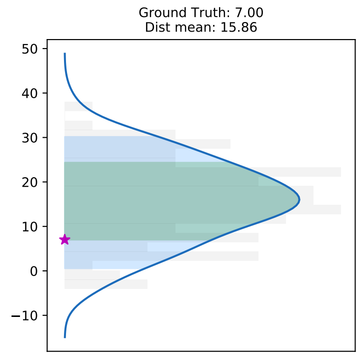
(a) Comparison between real and predicted RUL in CMAPSS FD004 data



(b)



(c)



(d)

Figure 5.18: RUL Prediction for JaNet architecture on FD004 CMAPSS dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 1, 125 and 245 respectively.

The progress of the loss function in this dataset is shown in figure 5.19 and has the same behavior as that previously analyzed in figure 5.13.

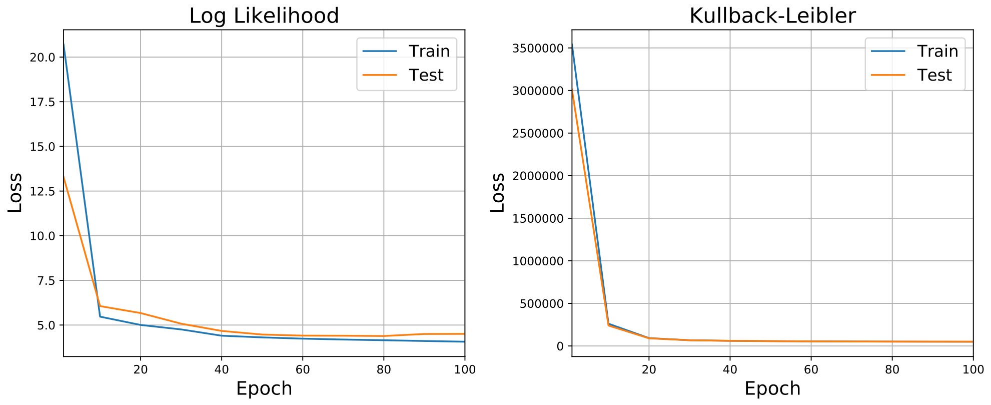


Figure 5.19: Log Likelihood and Kullback-Leibler in Bayes VRNN training for CMAPSS FD004 dataset.

5.1.5. Comparison Against different frequentist models

The first comparison is made against the same networks in their frequentist form, which means that the results are a comparison between the same architectures but with different weights approximations, in the case of frequentist networks, the weights are just an array of values while in the Bayesian networks the weights are sampled from normal distributions.

Table 5.6 shows the results for RMSE estimation. it is noticeable that in almost every case the Bayesian approach is better than the frequentist, even reaching differences of 20 % in VRNN with the most complex dataset.

Table 5.6: Results of Bayesian-RNN (mean RMSE) against frequentist RNN (RMSE) for RUL estimation in C-MAPSS datasets in three iterations each.

Model	VRNN		JaNet	
Dataset	Frequentist Model RMSE	Bayesian Model RMSE	Frequentist Model RMSE	Bayesian Model RMSE
FD001	14.62	14.28	15.02	15.06
FD002	21.15	16.78	20.58	18.44
FD003	16.88	13.86	16.65	12.70
FD004	24.71	20.11	24.68	21.95

The second comparison examines the performance of the proposed BRNN with weight perturbation and Bayes by Backprop approach against RNN based on Monte Carlo Dropout approach. The same architectures are applied, and MC Dropout is implemented with drop rate either 0.25 or 0.5. As shown in Table 5.7, the results demonstrate that the Bayesian RNNs architectures are capable of precisely predict the RUL of the turbofan and surpass the approach using MC Dropout in almost all kind of architecture.

Table 5.7: Results of Bayesian-RNN against the model that use MC Dropout as Bayesian approximation in three iterations each.

Model	VRNN			JaNet		
Dataset	Bayesian Model RMSE	MC-Dropout 0.5 RMSE	MC-Dropout 0.25 RMSE	Bayesian Model RMSE	MC-Dropout 0.5 RMSE	MC-Dropout 0.25 RMSE
FD001	14.28	19.95	15.44	15.6	15.16	14.90
FD002	16.78	20.82	21.04	18.44	21.41	21.56
FD003	13.86	21.15	22.75	12.70	20.37	17.43
FD004	20.11	26.32	25.05	21.95	25.44	24.62

The last comparison is with state-of-the-art models for CMAPSS dataset, to make this comparison, only the B-VRNN architecture is used since it is the one that presents the best results for all datasets. Since 2014 multiple neural networks models have been developed to predict RUL in CMAPSS dataset, some of the most successful attempts to complete this task are summarized below; Chong Zhang et al. [47] MODBNE which stands for Multi-Objective Deep Belief Networks Ensemble to generate RUL predictions, Xiang Li et al. [25] proposes a Deep Convolutional Neural Network with the same time window approach to the proposed model; the standard deviation is the deviation of many independent models. Jun Wu et al. [45] presents a Deep LSTM model for RUL prediction showing the improvement with deep fusion of sensory data and finally domain Adaptive CNN (Li, Li and He) [24] which can test not only with the test set of the same data but also with test sets from other data (i.e. train with FD001 and test with FD003).

Table 5.8: Results of Bayesian-RNN against state-of-the-art models

Model	MODBNE		DCNN		DLSTM		AdaBN-CNN		Bayesian VRNN	
Data	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
FD001	15.40	-	12.61	0.19	16.14	-	13.17	-	14.28	0.52
FD002	25.05	-	22.36	0.32	24.49	-	20.87	-	16.78	0.48
FD003	12.51	-	12.64	0.14	16.18	-	14.97	-	13.86	0.75
FD004	28.66	-	23.31	0.39	28.17	-	24.57	-	20.11	0.51

As it can be seen in table 5.8, the results obtained, in addition to being able to determine the uncertainty of the model, are also capable of presenting highly competent and even better results than most of the proposed models studied in this work.

The results obtained for the two more complex datasets are much better than any of the other models analyzed. As previously stated, this dataset is used to validate the proposed model, since the dataset is simulated, the next step is to use the models proposed in real datasets and demonstrate their efficiency.

The following sections, presents the results for the proposed approach in different datasets generated based on experiments.

5.2. RUL Prediction in Wind Turbine High Speed Bearings

The diagnosis of bearing health through the quantification of accelerometer data has been an area of interest for many years and has resulted in numerous signal processing methods and algorithms. In particular, bearing used in wind turbine generators (WTGs) are subjected to tough environments during operation, in the case of WTGs the bearing main objective is to support the loads in the rotor and the rotor shaft, this is why it is necessary to have a good indicator of the health state of the bearings.

In this section, a method to estimate the RUL inside of a WTG is presented, as the RUL is a continuous variable, this is a regression problem.

5.2.1. Data Description

This dataset is collected from a 2 $[MW]$ wind turbine high-speed shaft driven by a 20-tooth pinion gear. A vibrational signal of 6 seconds in a $100[kHz]$ sampling rate was acquired each day for 50 consecutive days. An inner race fault developed and caused the failure of the bearing across the 50-day period [40], the failure is shown on figure 5.20.

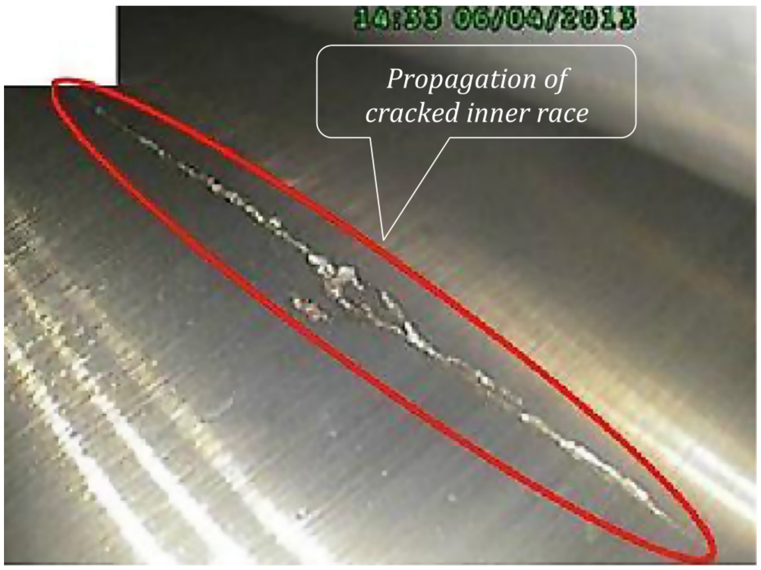


Figure 5.20: Real world HSS bearing fault: At the conclusion of the data collection, an inspection of the bearing revielsed a cracked inner race.

The data from a high speed shaft bearing from a WTG is provided by the Green Power Monitoring Systems in USA. An scheme for wind turbine generator is illustrated below:

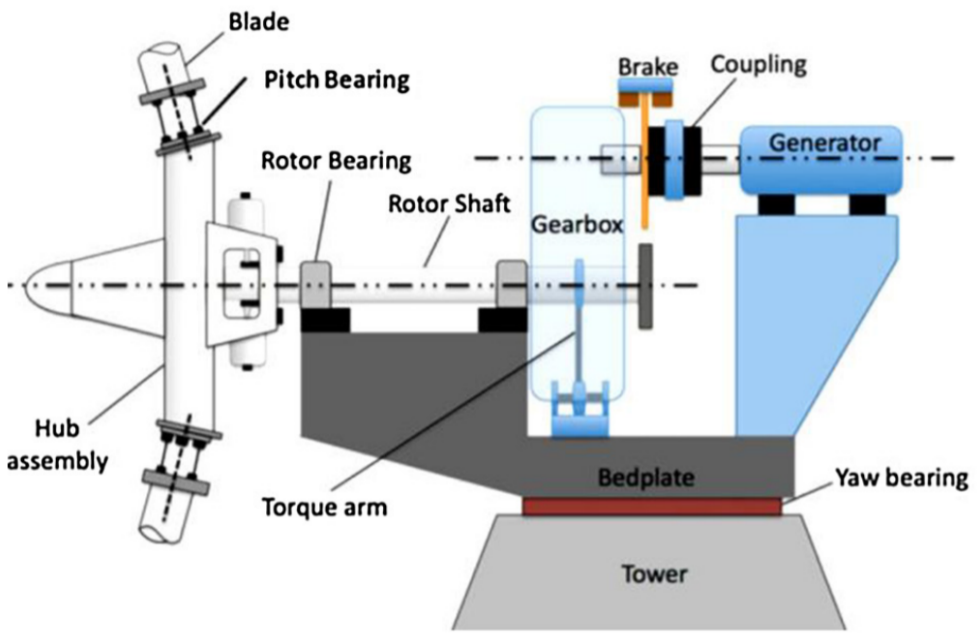


Figure 5.21: Components of the WTG including; tower; yaw system; generator; gearbox; low-speed shaft (main shaft); HSS; blades; nacelle; hub; meteorological unit (anemometry and wind vane); brake system; main bearing[27]

Bearings are typed SKF 32222 J2 tapered roller bearings. The TRB is 200 [mm] in outer diameter, with a bore of 110 [mm] and a total length of 56 [mm]. Note that the bearing under state also has a load cell to measure the force on the bearing. It's a variable speed control from 2 to 100 [Hz], with a load cell up to 1000 [lbs]. Most of the testing was done at 150 or 300 [lbs]. The highest test load applied was 50 % of rated power to reduce the chances of a catastrophic gearbox failure.

5.2.2. Data Preparation

To use this data in the model proposed for this work it is necessary to reorder the data for a recurrent neural network, this means that the order of the data must be *data-size, time-steps, features*, which means that every data point must consist in a temporally series with one or more features, to get more information about the signals, in this work a set of characteristics are obtained from the signal.

For the preprocessing in each file, it is necessary to *cut* the signal into batches to get features (*feature bacth*), after this process is applied on each file from every feature batch, it is possible to get all the different features mentioned below: mean, peak, peak to peak distance, crest factor, root mean square, variance, skewness, kurtosis, first five moments, and the first five band frequencies. Each of these features are calculated for the raw signal, the derivative from the signal and the integral of the signal, providing a total of 57 features for each feature batch, as this is a regression problem, for each feature batch the time to failure is registered and saved as is shown in figure below:

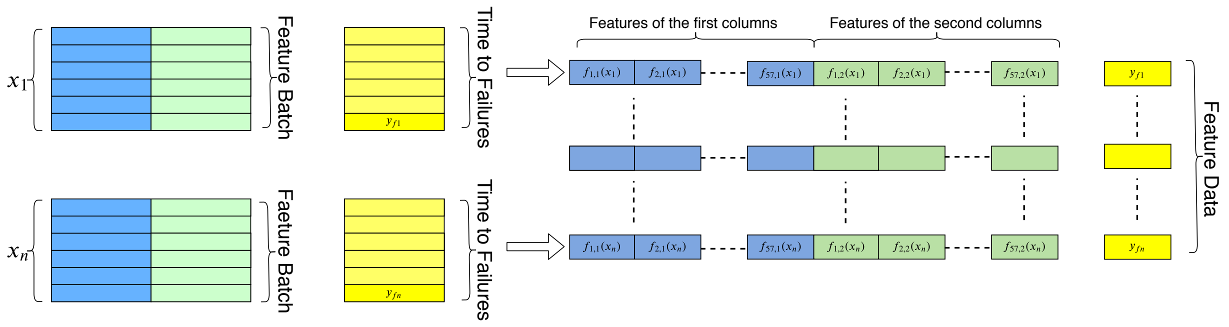


Figure 5.22: Preprocessing Scheme for *feature batches* in WTG data

where $f_{i,j}(x_k)$ is the feature i obtained in the column j of the data batch k, and y_{fi} is the time to failure of the features i.

After the dataset is transformed into the *feature dataset*, it is necessary to define a *time-step* to fully complete the requirements of the recurrent networks, for this, every data point between certain time interval is grouped into a temporal batch, with this, the data is able to be used into the proposed model.

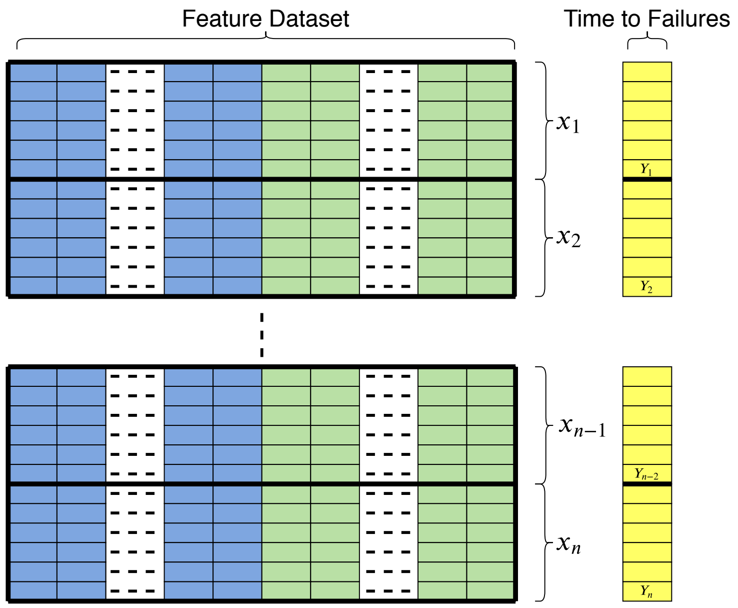


Figure 5.23: Scheme for the final step of preprocessing in WTG data

Where X_i is the data point i and Y_i is the time to failure of the *i-th* point. Finally, each point of the data consists in a time serie of the feature extracted from the original acceleration signal.

The *time-step* and *feature size batch* are chosen to obtain better results, the respective values of each are 200 for the feature batches and 50 for the entry of the recurrent neural networks, because the sampling frequency is $\approx$ 100 $[kHz]$ and each data point needs 10000 values, it can be said that each data point represents a time of 0.1 $[s]$, which means that for every data point, the quantity of revolutions of the bearings oscillates between 20 and 1000.

Before the data enters to the model it is randomly separated using a test-train ratio of 0.33, which means that the train set is the 66 % of the total data while the test set is the 33 % of the total data. After the data is divided, a scaler for the normalization of the data is trained with the training set and then using this scaler on the test set, the train set and the test set are normalized.

The indicator generated for this dataset is a health indicator that goes from 0 to 100 where 100 it means fully healthy and 0 means that the system failed, at the beginning of the preprocessing, every point is labeled with a health indicator according to their respective time to failure.

5.2.3. Results and Discussion

The results for the prediction of the health indicator of this dataset are presented below. The networks used are **Convolutional Bayesian RNN** (Conv-BayesRNN), this networks use an image as input, which means that the time series generated in subsection 5.2.2 are used as an image for the *Convolutional layers*. An important observation about this layers is that the kernel used is of the form $[1xi]$, because in this way the temporality of the time serie is not altered. After leaving the convolutional layers, the signal is operated by a *TimeDistributed layer*, which means that the output images of the convolutional layers are separated and flattened into time vectors and then enter the Bayesian Recurrent Layers. Finally, all the sequential data of the recurrent layers are flattened and used by the DenseFlipout layers to predict the bearing health indicator in the wind turbine. An outline of the networks used is shown below.

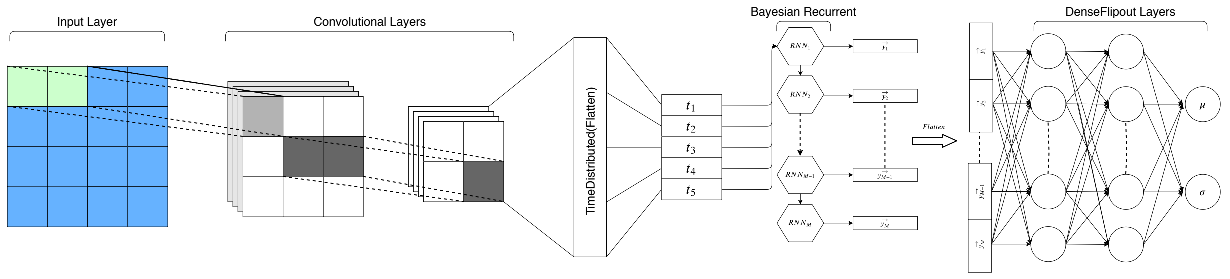


Figure 5.24: Scheme of RNN model used.

As seen in the previously illustrated figure, the main task of frequentist convolutional networks is to generate the *feature maps* as it was explained in subsection 2.3.2 so that these are then interpreted by recurrent Bayesian networks and finally the parameters of the output distribution are predicted by the DenseFlipout layers.

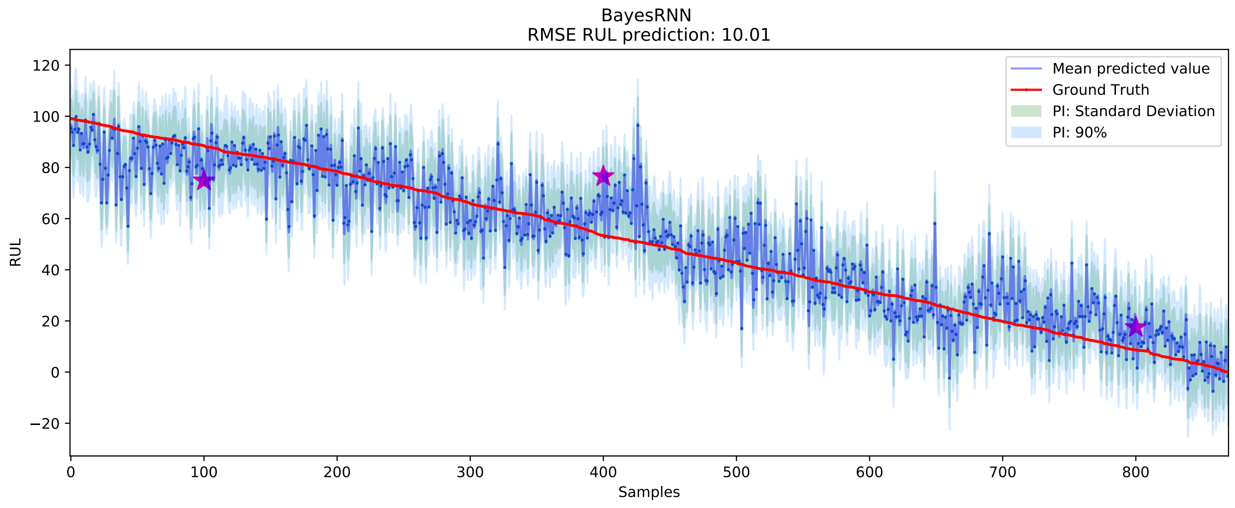
The number of epochs during which the model was able to obtain the most acceptable results is 3000, the KL divergence regulator has a value of $1e-7$, the net income batch size is 500 and finally the number of examples used to obtain the distributions for each example in the test set using the Monte Carlo method is 100.

- Bayes VRNN:
  The optimization process for the Conv-BayesRNN using VRNN as recurrent cell is done using a grid search. The final parameters used are shown in the table below:

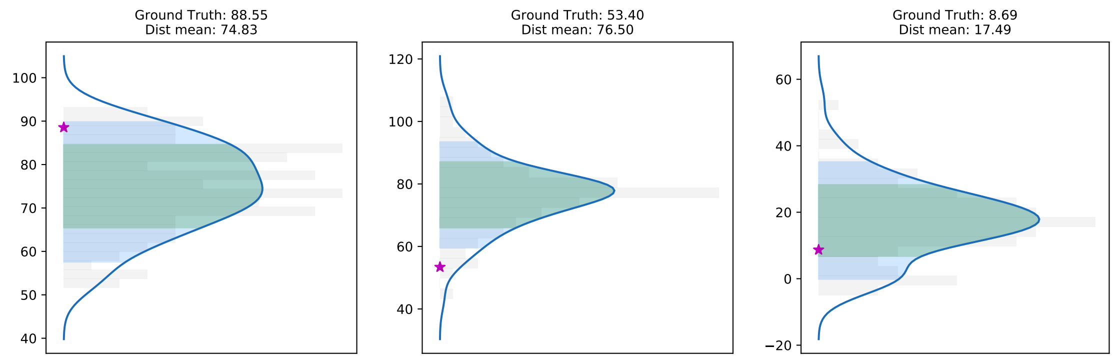
Table 5.9: Conv-Bayes VRNN parameters

Layer	Neurons/Units/Filters	Kernel	Activation Function
Conv2D	32	[1,2]	ReLu
Conv2D	64	[1,2]	ReLu
BayesVRNN	64	-	ReLu
DenseFlipout	16	-	ReLu
DenseFlipout	2	-	

The results for the health indicator prediction are shown in figure 5.25, in which it can be seen that the proposed model is capable of identifying and accurately predict the bearing health indicator, it is important to note that the standard deviation in almost all the test examples involves the actual result, which is a good indicator to say that the model is capable of quantifying uncertainty in an accurate way.



(a) Comparison between real and predicted RUL in WTG data



(b) (c) (d)

Figure 5.25: RUL Prediction for VRNN architecture on WTG dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 100, 400 and 800 respectively.

As seen in figure 5.26, the behavior of the loss function is as expected, that is, both functions stabilize after 6000 epochs and this means that there is a correct regularization of the network, together with a good quantification of the model uncertainty .

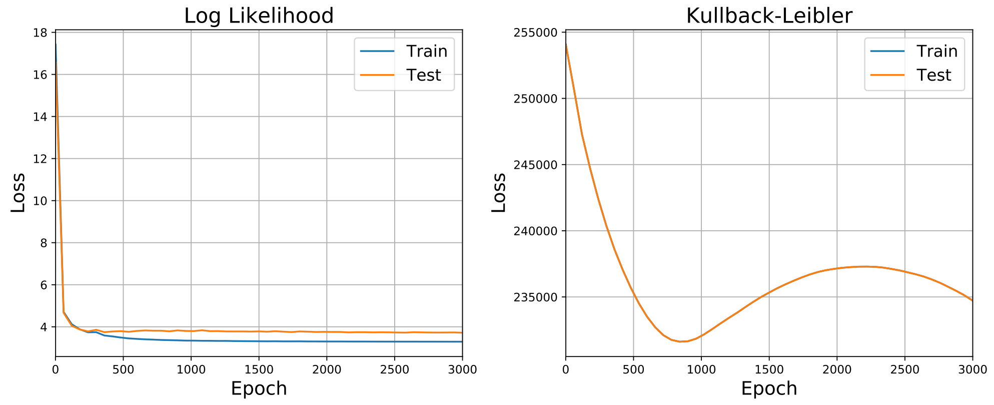


Figure 5.26: Log Likelihood and Kullback-Leibler in Bayes VRNN training for WTG dataset.

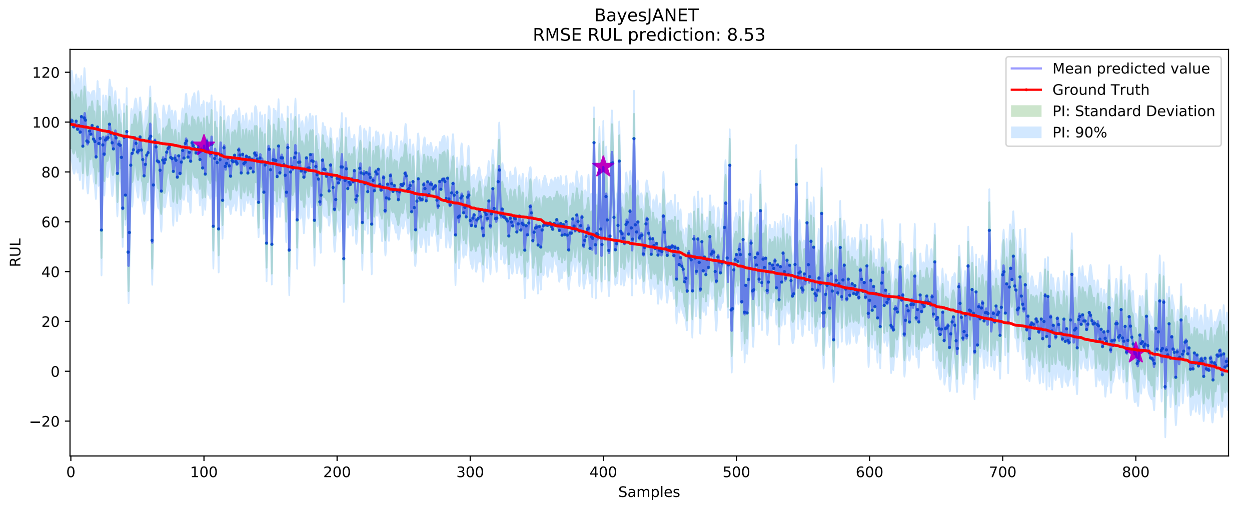
As seen in the previous figure, there is a slight overfit at the time of training, however, this is not significant considering the results finally obtained with this model.

- Bayes JaNet: Below, are the optimized parameters for the Conv-Bayes JaNet network in table 5.10, the network scheme is the same as in figure 5.24.

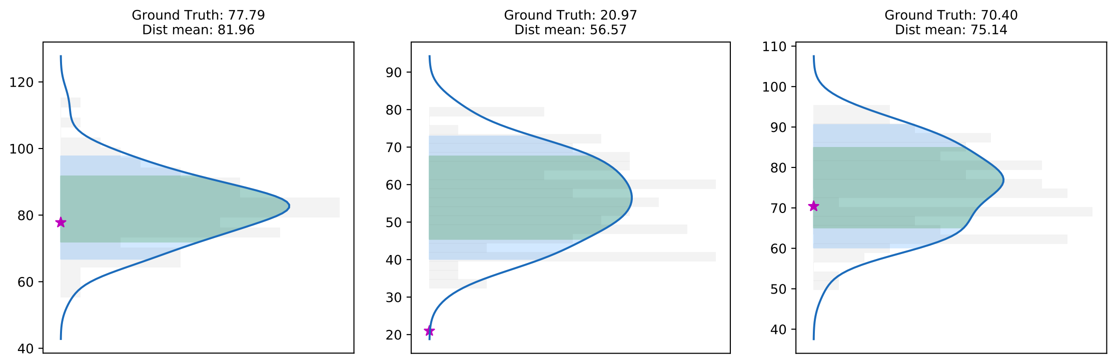
Table 5.10: Conv-Bayes VRNN parameters

Layer	Neurons/Units/Filters	Kernel	Activation Function
Conv2D	32	[1,2]	ReLu
Conv2D	64	[1,2]	ReLu
Bayes JaNet	64	-	ReLu
DenseFlipout	16	-	ReLu
DenseFlipout	2	-	

As seen in figure 5.27, the results for this architecture are better than those previously presented for the Conv-Bayes VRNN network, however, in both cases it can be seen that the model is able to accurately predict bearing degradation by delivering. Thus, a health indicator directly related with the operational condition of the bearing is predicted by the developed models.



(a)



(b) (c) (d)

Figure 5.27: RUL Prediction for JaNet architecture on WTG dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distribution predicted for the sample 100, 400 and 500 respectively.

In figure 5.28 it can be seen that the behavior is similar to the one seen previously in figure 5.26, so the pertinent observations are explained in the previous item.

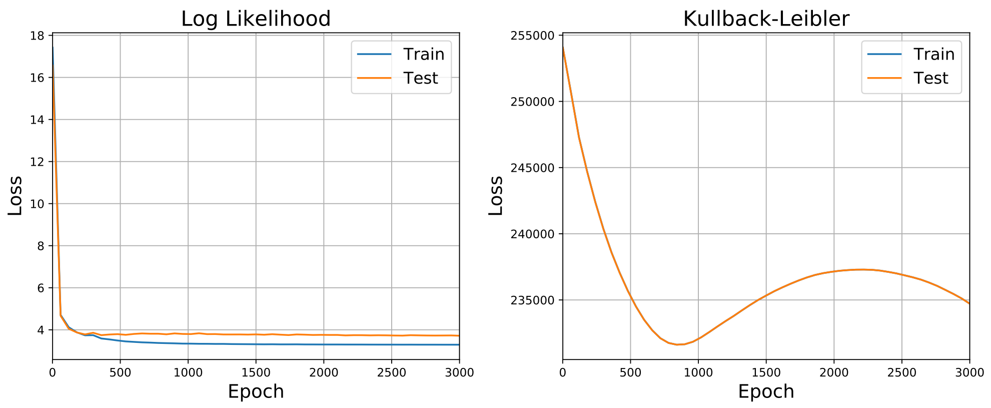


Figure 5.28: Log Likelihood and Kullback-Leibler in Bayes VRNN training for WTG dataset.

It is important to emphasize that given the nature of the data, where it considers only one failure, the separation between the train-test sets is random with a radius of 33 % as explained above. This, however, could contemplate certain complications, since the model is capable of predicting behavior only for this *type* of failure, it is possible that under different conditions the model cannot predict an accurate health indicator, however, to generalize it, a larger set of failures are required.

A summary with the RMSE results obtained for three iterations of each of the developed models is presented below in table 5.11.

Table 5.11: RMSE for all the iterations in WTG dataset.

Iteration	Conv Bayes-VRNN	Conv Bayes-JaNet
1	10.25	8.53
2	10.97	9.11
3	10.01	9.26
Mean (STD)	10.41 (0.40)	8.96 (0.31)

5.3. Health Indicator Prediction in Cracks

For this dataset, the main objective is to determine the remaining useful life of metal parts subjected to cyclic loads. This dataset is obtained through a series of experiments developed by the University of Maryland.

5.3.1. Data Description

The experimental processes developed consist of the collection of three dissipative energies; deformation energy, heat dissipation and acoustic emission. Each dissipative energy data was converted to damage-representing entropy. Particularly, classical thermodynamic entropy is used as the reference damage measure and used to assess the performance of the other two entropic approaches.[46]

Crack length and cycle life measurements are used to determine the moment of failure, the failure point in crack length is determined differently at initiation, 250 μm crack, 500 μm crack, 1000 μm crack, the transition from region II to III (following linear elastic fracture mechanics (LEFM) [1]), and fracture. These failure determinations are used in each entropic approach's assessment [46].

The mechanical and chemical properties of the specimen used for this dataset is shown below in table 5.12. The dogbone-shape for the specimen was selected and designed for fatigue testing under the ASTM 466 guideline [11].

Table 5.12: Mechanical properties and chemical composition of specimen material SS304L.

Mechanical Properties									
σ_{uts}		σ_{yield}		Elongation[%]			Hardness[RB]		
613.8		325.7		54.06			85.00		
Chemical Composition									
C	Cr	Cu	Mn	Mo	N	Ni	P	S	Si
0.024	18.06	0.366	1.77	0.294	0.071	8.08	0.030	0.001	0.193

In the uniaxial loading test, a servo-hydraulic testing system was used. An Instron 8800 system retrofitted on an MTS 311.11 frame, the specimens are clamped from its upper and lower end, at the lower end the actuator is connected so the cyclical forces are applied at this point. The loading conditions belong to the range 9 $\sim$ 15 [kN], the stress ratio corresponds to 0.1 and the frequency is 5 [Hz]. Every 1000 cycles the load is paused and 500 excitation cycles are applied at 25 [Hz] and 6 [kN]. After each period of charge and excitation, the sample remains at rest for a period of two minutes to allow the sample to cool and the measurements are not disturbed.

The loading and extension data was collected by the Instron 8800 system, the acoustic data was collected from two Physical Acoustics Micro-30s resonant sensors and the crack length was measured by a microscope system (Edmond 2.5-10X microscope body combined with OptixCam Pinnacle Series CCD digital camera).

5.3.2. Data Preparation

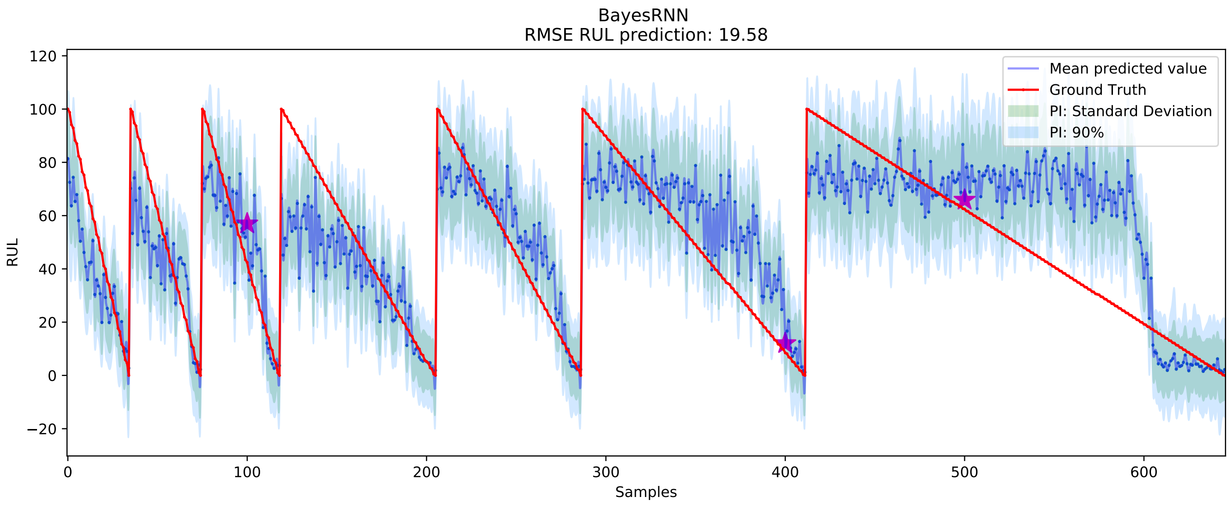
The preprocessing used for this dataset is similar to the one seen previously for the CMAPSS dataset, that is to say, for this dataset characteristics were extracted for each of the measurements. This means that the remnant useful life of the last time-step is used to generate the labels of the temporary window, as it can be seen in figure 5.2. However, since there are very high remaining useful life times, for this dataset each remaining useful life time is normalized with respect to the fault to which it belongs, thus obtaining a health indicator that varies between zero and one hundred, where one hundred means that the equipment is in an optimal state of health and zero corresponds to the failure.

In order to not have test-train contamination, seven random failures are extracted from the dataset to be used to test the model, while the others are used for training, so in a real case, the model will be able to predict the life time useful of a sample under the same experimental conditions.

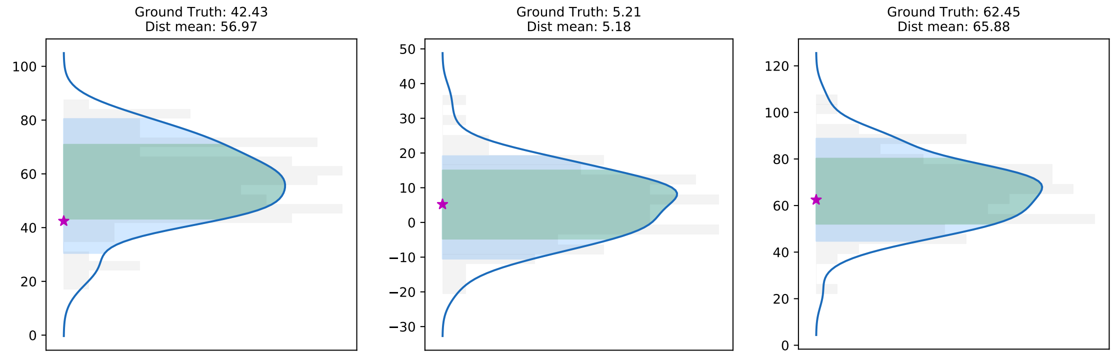
5.3.3. Results and Discussion

The results for the two architectures studied in this work are presented below. The parameters for the optimizations of both networks were obtained through a grid search, the time necessary for the stabilization of the models is 2000, the batch size is 1000, the number of samples used to obtain the output distributions is 100, and the KL divergence regulator is $1 - 7$.

- Bayes VRNN: The results obtained for the VRNN architecture are presented in figure 5.29 in which it can be seen that the network is capable of predicting the behavior of all failures independently, however, in the last failure the network is not capable of correctly determine the point of degradation, this may be due to the simplification of the labeling process or to a lack of optimization in the model.



(a) Comparison between real and predicted health indicator in Maryland Cracks data



(b) (c) (d)

Figure 5.29: Health Indicator Prediction for VRNN architecture on Maryland Cracks dataset, image (a) is a full comparison between the real and predicted Health Indicator, while figures (b), (c) and (d) are the probability distribution predicted for the sample 100, 200 and 300 respectively.

The network training process is visualized in figure 5.29 in which the behavior turns out to be as expected, ending with a stabilization of the log-likelihood and a decreasing behavior in the KL divergence, which means that the proposed distribution approaches the prior while the predicted values approach the actual values..

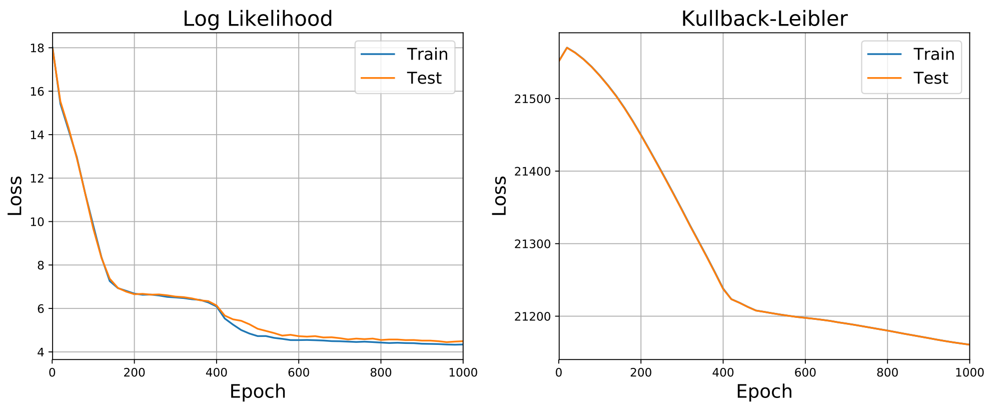
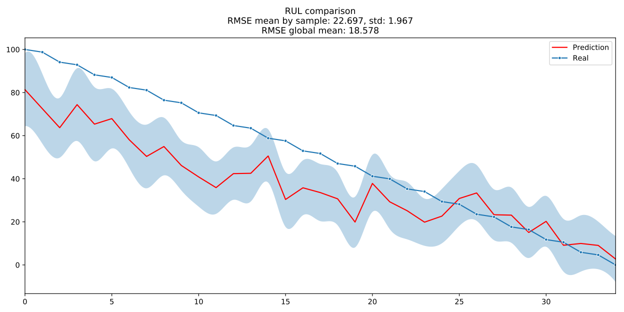
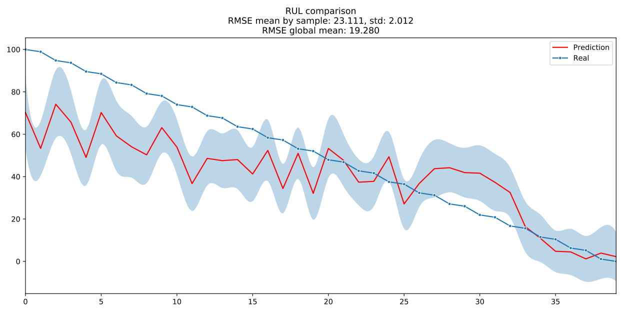


Figure 5.30: Log Likelihood and Kullback-Leibler in Bayes VRNN training for Cracks dataset.

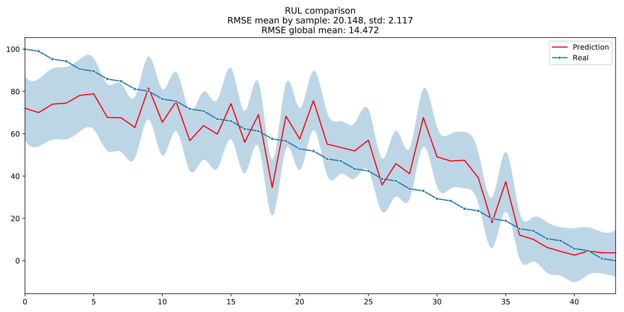
The figures below show detailed images of the predicted health indicator in the test set, in these figures, it can be distinguished the predictive capacity of the health indicator of the proposed model.



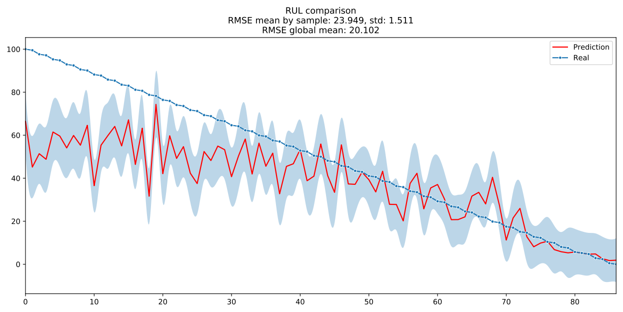
(a)



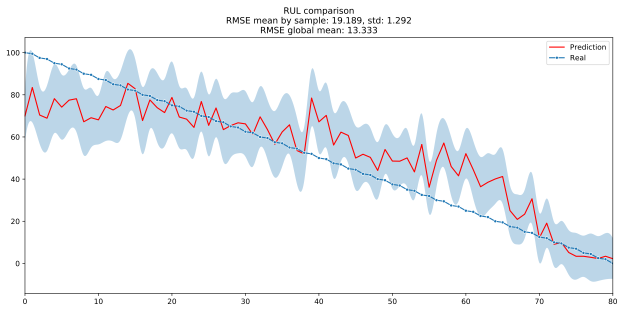
(b)



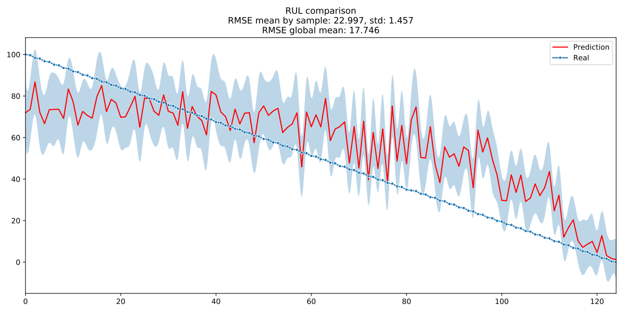
(c)



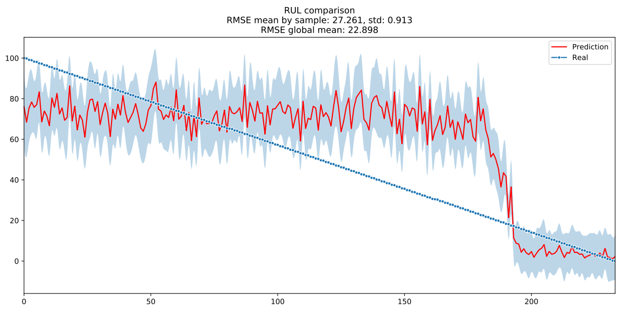
(d)



(e)



(f)

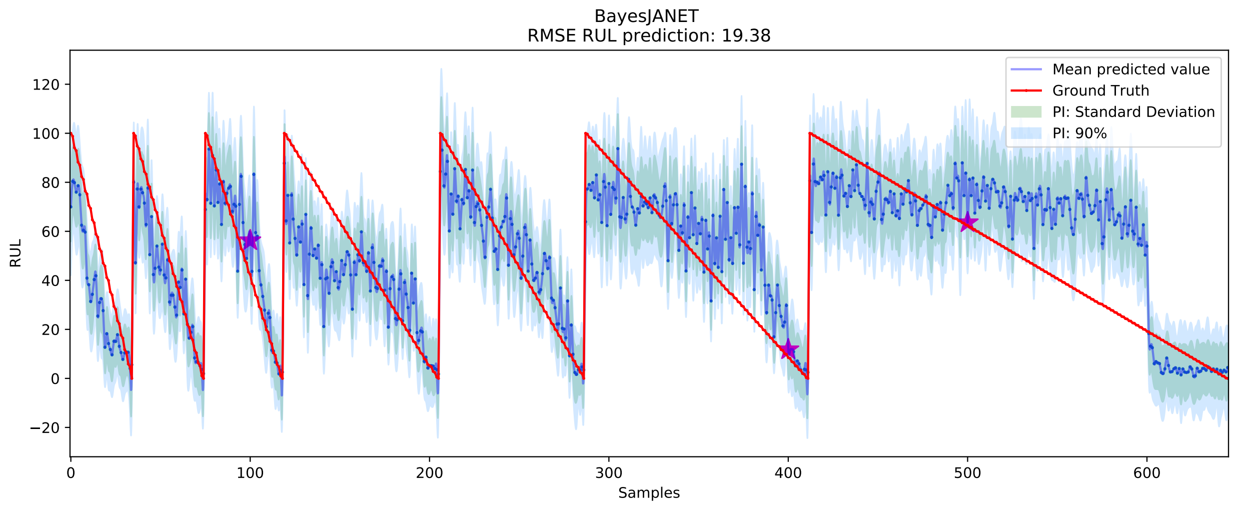


(g)

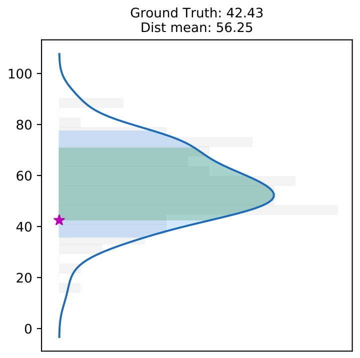
Figure 5.31: Health indicatior prediction for every sample in test set for VRNN architecture on Maryland Cracks dataset.

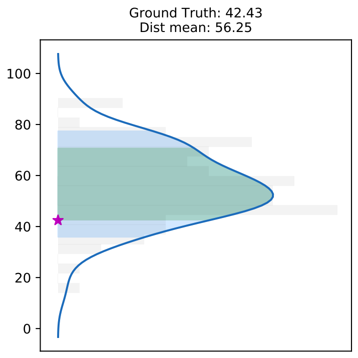
As can be seen in the previously illustrated figures, the health indicator obtained by the VRNN architecture is able to distinguish the degradation process in the cracks, however, in some cracks the predictions are far from the real value. This may be mainly due to unforeseen failures in the sample corresponding to the crack studied.s.

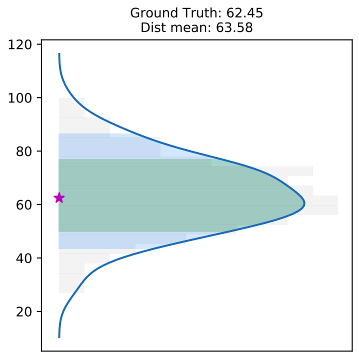
- Bayes JaNet: The results obtained for the architecture corresponding to the JaNet cells are illustrated below, in figure 5.32. The model proposed, like in the previous item, is capable of predicting the behavior of the dataset, which means that this model can predict the Health Indicator of the sample.



(a)







(b) (c) (d)

Figure 5.32: Health indicator prediction for JaNet architecture on Maryland Cracks dataset, image (a) is a full comparison between the real and predicted RUL, while figures (b), (c) and (d) are the probability distributions predicted for the sample 100, 200 and 300 respectively.

The behavior of the cost function is the same as in the previous case. This can be seen in figure 5.33.

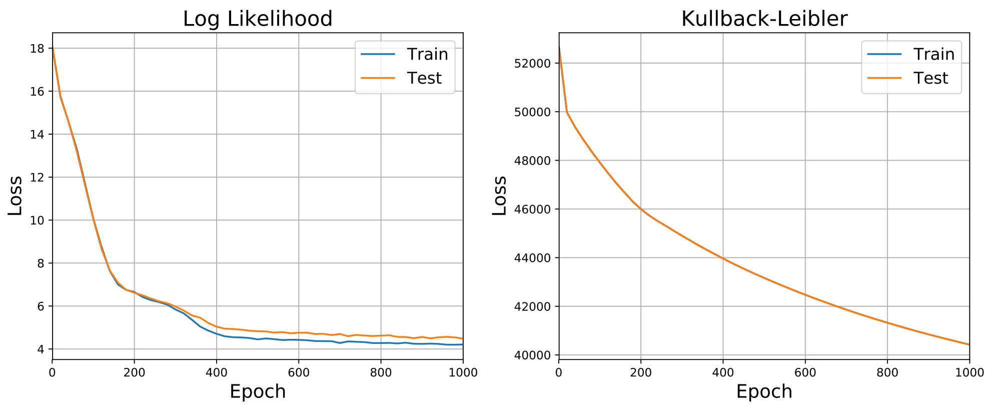
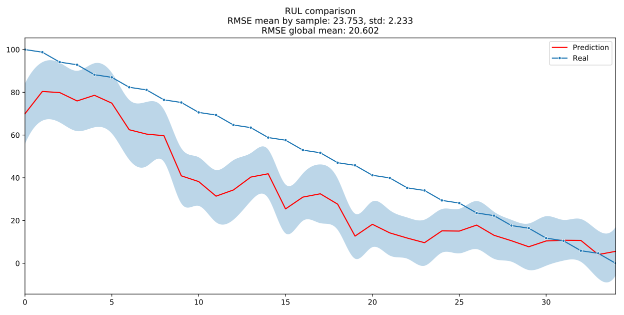
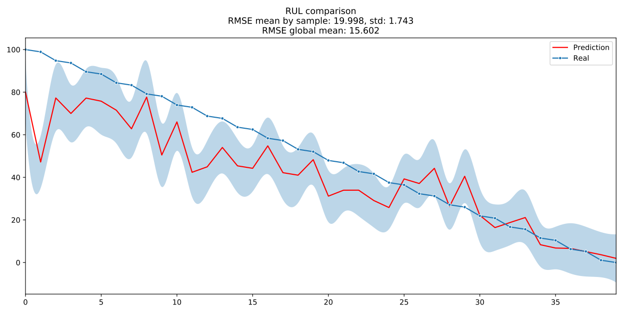


Figure 5.33: Log Likelihood and Kullback-Leibler in Bayes VRNN training for Cracks dataset.

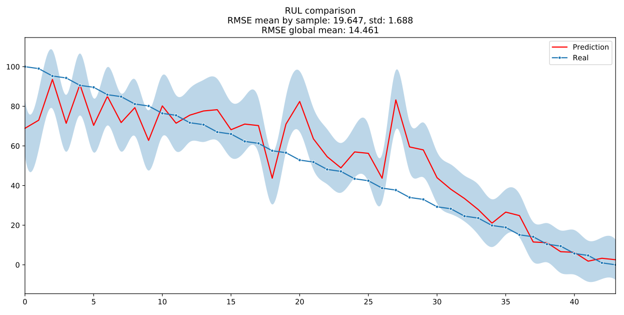
A detail of the failures studied in the test set is presented below, the results for the prediction of low Health Indicator is much better than for the larger ones, again, this is due to the number of samples present in the training set that have similar characteristics.



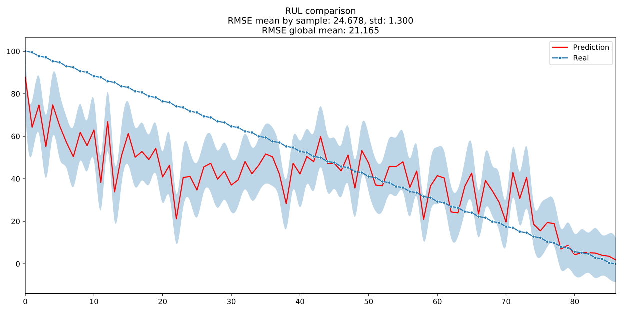
(a)



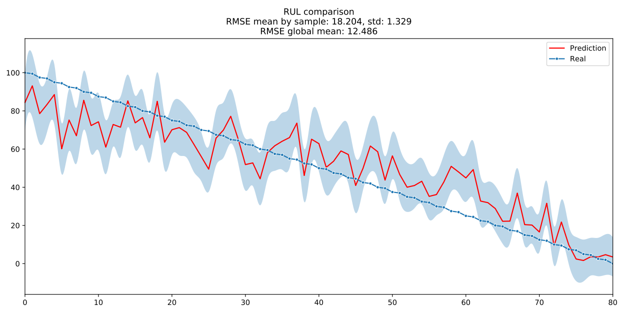
(b)



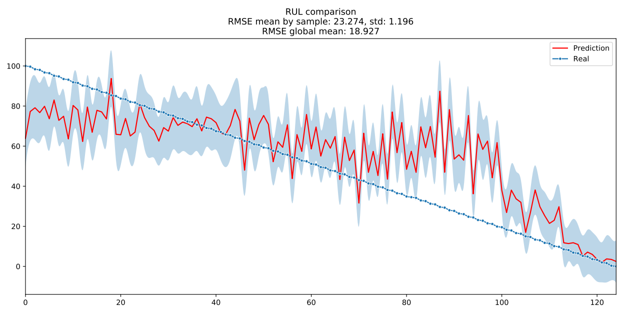
(c)



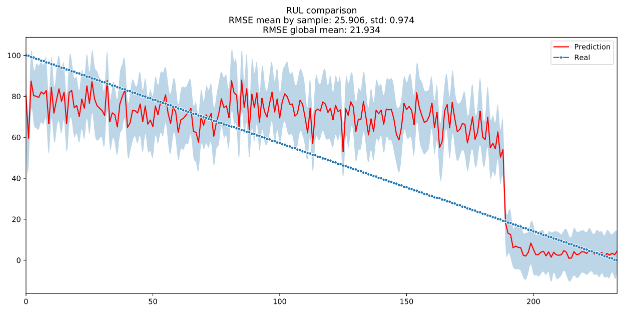
(d)



(e)



(f)



(g)

Figure 5.34: Health indicator prediction for every sample in test set for JaNet architecture on Maryland Cracks dataset.

The errors in the predictions of the Bayes JaNet architecture are similar to those studied for the previous model, and are mainly due to unexpected failures within the samples studied.

A summary with the RMSE results obtained for three iterations of each of the developed models is presented below in Table 5.13.

Table 5.13: RMSE for all the iterations in Maryland Cracks dataset.

Iteration	Bayes-VRNN	Bayes-JaNet
1	20.20	19.64
2	20.20	19.38
3	19.58	19.32
Mean (STD)	19.99 (0.29)	19.45 (0.18)

5.4. Health State Diagnosis Bearings: University of Ottawa

In this section, free access dataset of vibrating bearings of Univesity of Ottawa is studied. The fist part of this section explains the content of the dataset followed by a section about the preproccesing of the data. Finally, the results and discussion of the prediction of the health state of the data is presented.

5.4.1. Data Description

The experiments are performed on a SpectraQuest machinery fault simulator (MFS-PK5M) ilustrated in figure 5.35. The shaft is driven by a motor and the rotational speed is controlled by an AC drive. Two ER16K ball bearings are installed to support the shaft, the left one is a healthy bearing and the right one is the experimental bearing, which is replaced by bearings of different health conditions [18].

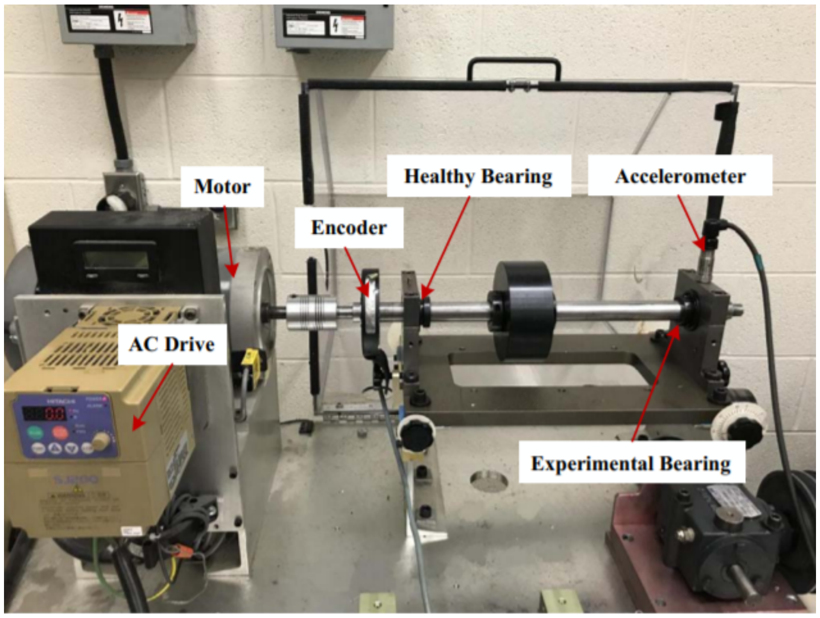


Figure 5.35: Experimental Setup for the vibrational data of University of Ottawa.[18]

The sensors located at the set up are an accelerometer (ICP accelerometer, Model 623C01) and an incremental encoder (EPC model 775).

In the experiments, two signals are obtained for the sensors, and both signals are sampled at 200.000 Hz in 10 second periods. The final dataset consists in 36 experiments, for each one there are two experiment settings: the bearing health condition, and varying the speed condition. The health conditions includes healthy, faulty with an inner race defect, faulty with an outer race defect, and multiple failures. The operating rotational speed condition are increasing speed, decreasing speed, increasing, then decreasing speed and decreasing, then increasing speed.

Table 5.14: Parameters of Bearings

Bearing type	Pitch diameter	Ball diameter	Number of balls	BPFI	BPFO
ER16K	38.52 mm	7.94 mm	9	5.43fr	3.57fr

The main problem in PHM context is to diagnose the health state of the bearings, this means that the problem is a classification type problem, and the classes in this case are the five health states explained above.

Table 5.15: Dataset Format

Bearing Health conditions	Speed Varying Conditions			
	Increasing speed	Decreasing speed	Increasing then decreasing speed	Decreasing then increasing speed
Healthy	H-A-1 H-A-2 H-A-3	H-B-1 H-B-2 H-B-3	H-C-1 H-C-2 H-C-3	H-D-1 H-D-2 H-D-3
Faulty (inner race fault)	I-A-1 I-A-2 I-A-3	I-B-1 I-B-2 I-B-3	I-C-1 I-C-2 I-C-3	I-D-1 I-D-2 I-D-3
Faulty (outter race fault)	O-A-1 O-A-2 O-A-3	O-B-1 O-B-2 O-B-3	O-C-1 O-C-2 O-C-3	O-D-1 O-D-2 O-D-3
Faulty (ball fault)	B-A-1 B-A-2 B-A-3	B-B-1 B-B-2 B-B-3	B-C-1 B-C-2 B-C-3	B-D-1 B-D-2 B-D-3
Faulty (combinated faults)	C-A-1 C-A-2 C-A-3	C-B-1 C-B-2 C-B-3	C-C-1 C-C-2 C-C-3	C-D-1 C-D-2 C-D-3

5.4.2. Data Preparation

To use this data in the model proposed for this work, it is necessary to reshape the data for a recurrent neural network, this means that the order of the data must be *data-size, time-steps, features*, which means that every data point must consist in a temporally serie with one or more features, to get more information about the signals, in this work a set of characteristics are obtained from the signal.

For the preprocessing of this dataset, the first step is to separate the data for sample, which means that every file contains 10 seconds of bearing measurements in one operational condition, for each file it is necessary to *cut* the signal into batches to get features (*feature bacth*). After this process is applied on each file from every feature batch, it is possible to get all the different features mentioned below: mean, peak, peak to peak distance, crest factor, root mean square, variance, skewness, kurtosis, first five moments, and the first five band frequencies. Each of these features are calculated for the raw signal, the derivative from the signal and the integral of the signal, providing a total of 57 features for each signal. A scheme of this process is shown in figure 5.36.

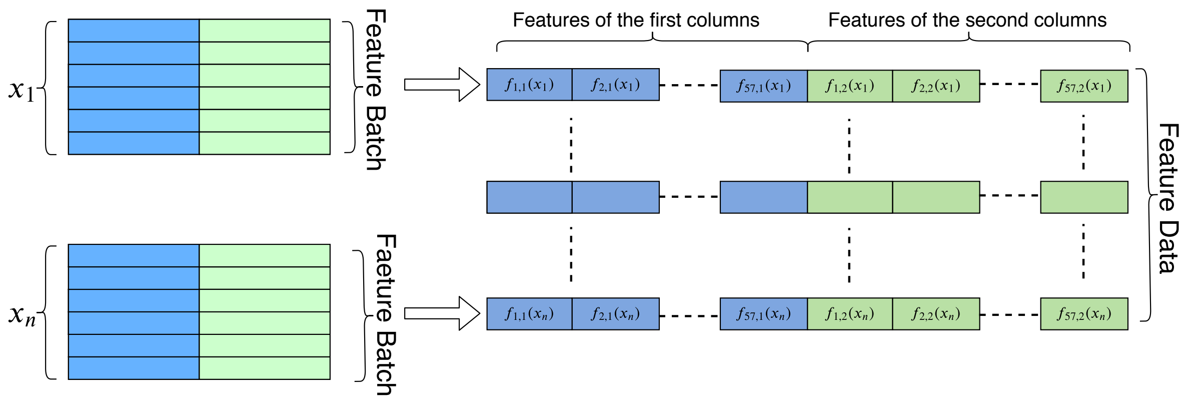


Figure 5.36: Preprocessing Scheme for *feature batches* in Ottawa Bearings

where $f_{i,j}(x_k)$ is the feature i obtained in the column j of the data batch k. It is also important to note that this process of feature extraction is useful in this case because of the large amount of data, however, the data reduction caused by the application of this method is not negligible.

After the dataset is transformed into the *feature dataset*, it is necessary to define a *time-step* to fully complete the requirements of the recurrent networks, for this, every data point between certain time interval is grouped into a temporal batch, with this, the data is able to be used in the proposed model.

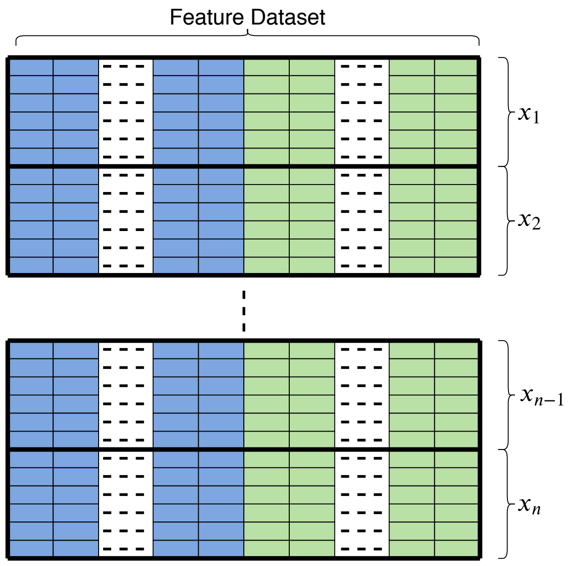


Figure 5.37: Scheme for the final step of preprocessing od WTG Data

The selection of the size of the feature batch and the time-step size it is not a random choice, these values are calculated based on the minimum rotation speed of the bearings, that is, the total size of each temporary batch is such that the bearing with the lowest rotation speed is capable of advancing half a revolution, the final value for both values is 50 data points, this means that 2500 data points are necessary to get one temporary batch, this is $0.0125[s]$.

Summarizing, to work this dataset, two processes are applied to the data, and in both of them the amount of data is reduced by the size of the feature batch, and by the size of the time-step, specifically, the initial amount of data is 4000000 data points, after the first process of feature batches, the remaining amount of data point is 80000, and finally after the preprocess for recurrent networks, the final amount of data points is 16000.

To avoid train-test contamination, the files corresponding to the health states and operating conditions remained separate in each process, this is important because in this way, the labels of each point remains in the file and tagging files doesn't become a problem. Since there were three tests for each operating condition, the final dataset was separated into three, where two of the three parts of the dataset were used to train the models, and the last one was used to test the model, this process was done in such a way that the three tests of the bearings could be evaluated in the testing phase of the model.

5.4.3. Results and Discussion

This problem corresponds to a classification task, so the trained distribution is not the same as the one trained previously, the distribution in this case is a categorical distribution, which means that the number of neurons in the last layer is equal to the number of classes of the problem, in this particular problem the categories are five, the structure of the network is showed below:

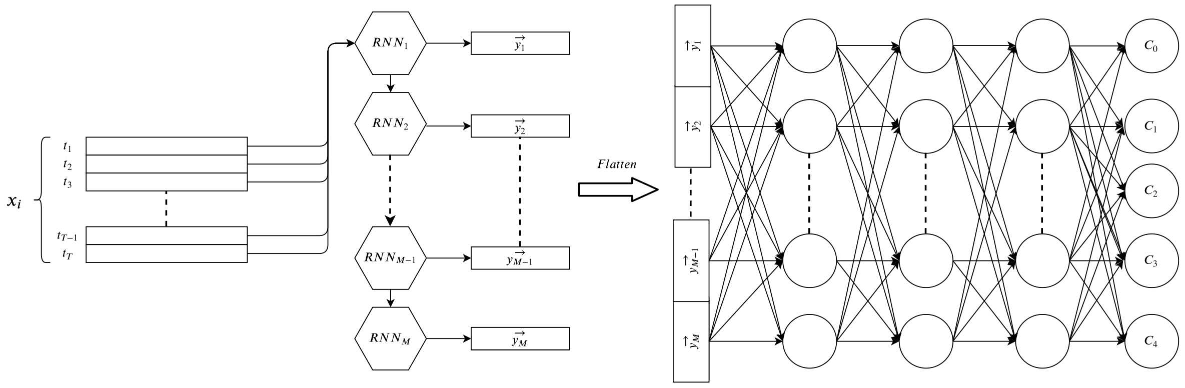


Figure 5.38: Scheme of a Bayesian RNN for classification task.

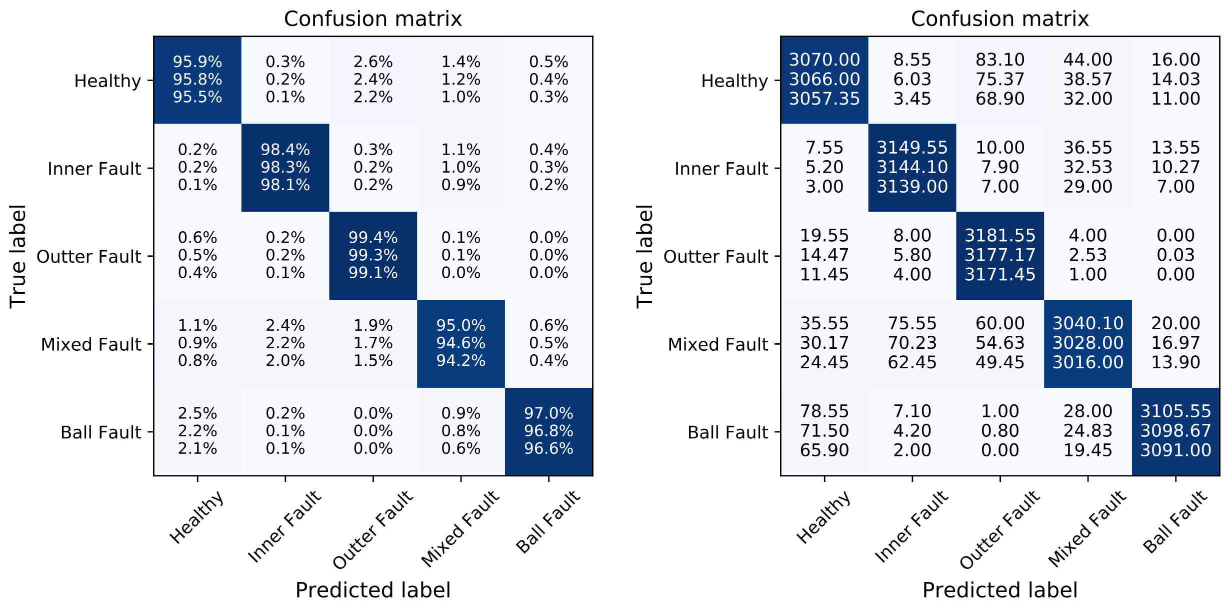
The same experiments are made for both recurrent cells, Bayesian Vanilla RNN and Bayesian JaNet, the results are presented below:

- Bayes VRNN:
  The optimization process for Bayes RNN using VRNN as recurrent cell is done via grid search, the final parameters used are shown in the table below:

Table 5.16: Parameters for Bayesian VRNN in Ottawa Bearings Datset.

Layers	Units/Neurons	Activation Function
VRNN	64	Relu
DenseFlipout	64	Relu
DenseFlipout	5	

The main result for a classification process is the confusion matrix and the accuracy. As mentioned above, this dataset was separated according to the three tests for each of the operating conditions, however, to avoid bias in the results, each model was iterated three times. The results of an iteration are presented below.



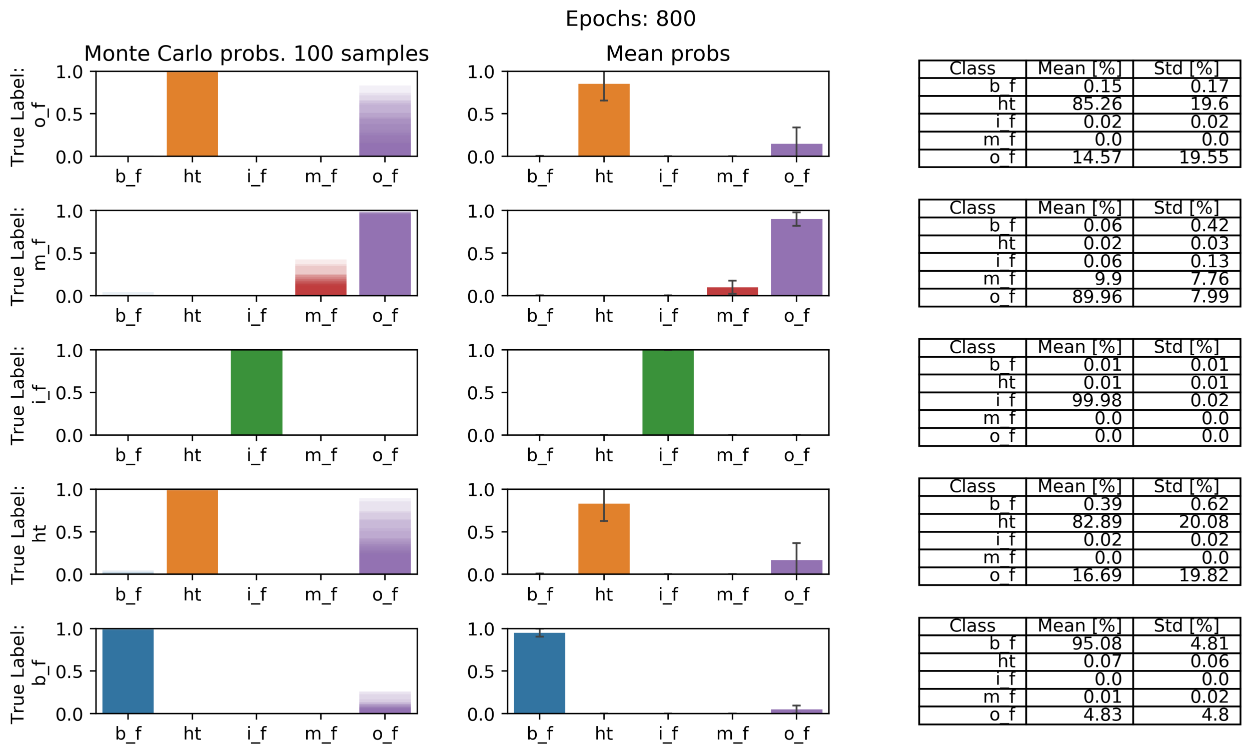
(a) (b)

Figure 5.39: Confusion Matrix 5.39b and normalized confusion matrix 5.39a for the test-set in Ottawa Bearing Dataset using VRNN architecture.

Where b_f means ball fault, ht means healthy, i_f means inner fault, m_f means multiple faults and o_t means outer fault.

As it can be seen in figure 5.39 the accuracy for this iteration is over 97 % ,furthermore, it can be seen that in all the categories the accuracy is very similar, so it could be said that the proposed model is able to generalize the structure of the data and thus, accurately correlating it with a category.

In the case of a classification task, the visualization of the uncertainty of the model is not as direct as in the case of a regression, in order to visualize it in this case, a sample of the test set was extracted for a value of each class chosen randomly. As it can be seen in figure 5.40, the model does not always classify data with the same category, this is because in some cases, the network is not robust enough to choose a single category for a sample.



Class	Mean [%]	Std [%]
b_f	0.15	0.17
ht	85.26	19.6
i_f	0.02	0.02
m_f	0.0	0.0
o_f	14.57	19.55

Class	Mean [%]	Std [%]
b_f	0.06	0.42
ht	0.02	0.03
i_f	0.06	0.13
m_f	9.9	7.76
o_f	89.96	7.99

Class	Mean [%]	Std [%]
b_f	0.01	0.01
ht	0.01	0.01
i_f	99.98	0.02
m_f	0.0	0.0
o_f	0.0	0.0

Class	Mean [%]	Std [%]
b_f	0.39	0.62
ht	82.89	20.08
i_f	0.02	0.02
m_f	0.0	0.0
o_f	16.69	19.82

Class	Mean [%]	Std [%]
b_f	95.08	4.81
ht	0.07	0.06
i_f	0.0	0.0
m_f	0.01	0.02
o_f	4.83	4.8

Figure 5.40: Distribution for the results in VRNN model for Ottawa Bearing dataset.

The evolution of the loss function for the training of this network is illustrated in figure 5.41 in which it can be seen that both terms in the cost function decrease, which means that the model is properly regulated and that it converged.

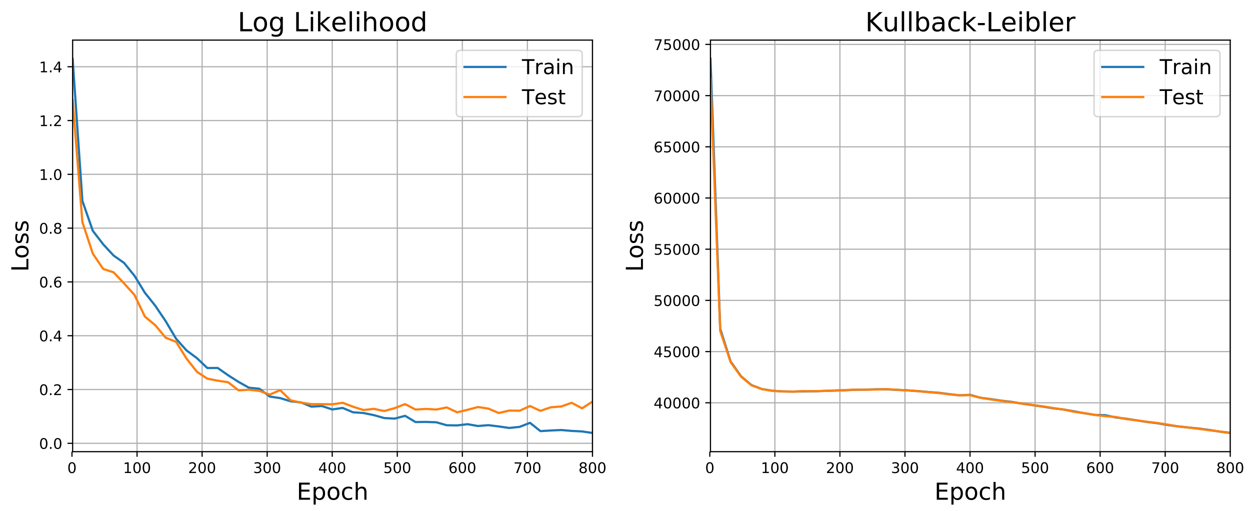
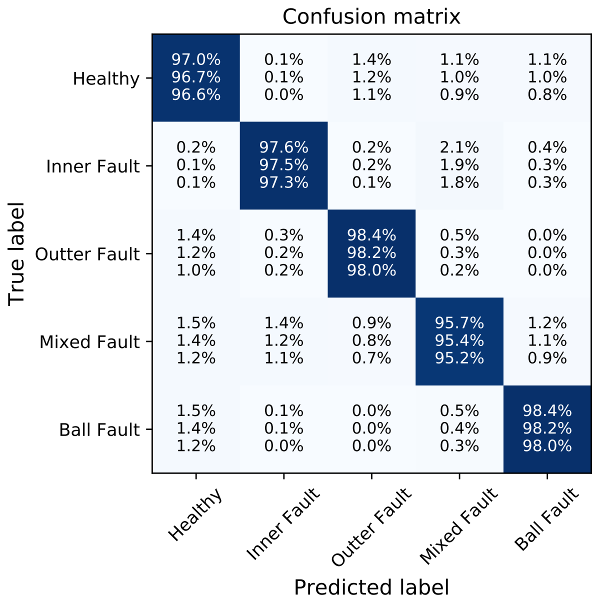


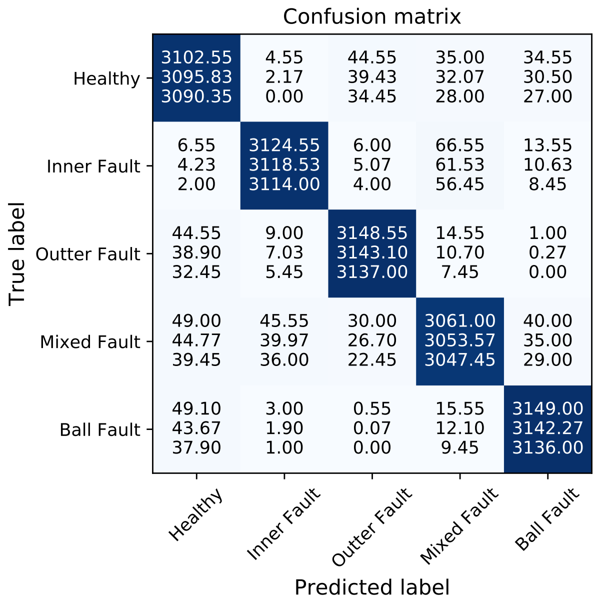
Figure 5.41: Log Likelihood and Kullback-Leibler in Bayes VRNN training for Ottawa Dataset.

- Bayes JaNet:
  The parameters used for this network are similar to those presented in table 5.16 and were obtained through a grid search, the confusion matrices for this architecture are presented below.



True label \ Predicted label	Healthy	Inner Fault	Outter Fault	Mixed Fault	Ball Fault
Healthy	97.0% 96.7% 96.6%	0.1% 0.1% 0.0%	1.4% 1.2% 1.1%	1.1% 1.0% 0.9%	1.1% 1.0% 0.8%
Inner Fault	0.2% 0.1% 0.1%	97.6% 97.5% 97.3%	0.2% 0.2% 0.1%	2.1% 1.9% 1.8%	0.4% 0.3% 0.3%
Outter Fault	1.4% 1.2% 1.0%	0.3% 0.2% 0.2%	98.4% 98.2% 98.0%	0.5% 0.3% 0.2%	0.0% 0.0% 0.0%
Mixed Fault	1.5% 1.4% 1.2%	1.4% 1.2% 1.1%	0.9% 0.8% 0.7%	95.7% 95.4% 95.2%	1.2% 1.1% 0.9%
Ball Fault	1.5% 1.4% 1.2%	0.1% 0.1% 0.0%	0.0% 0.0% 0.0%	0.5% 0.4% 0.3%	98.4% 98.2% 98.0%

(a)

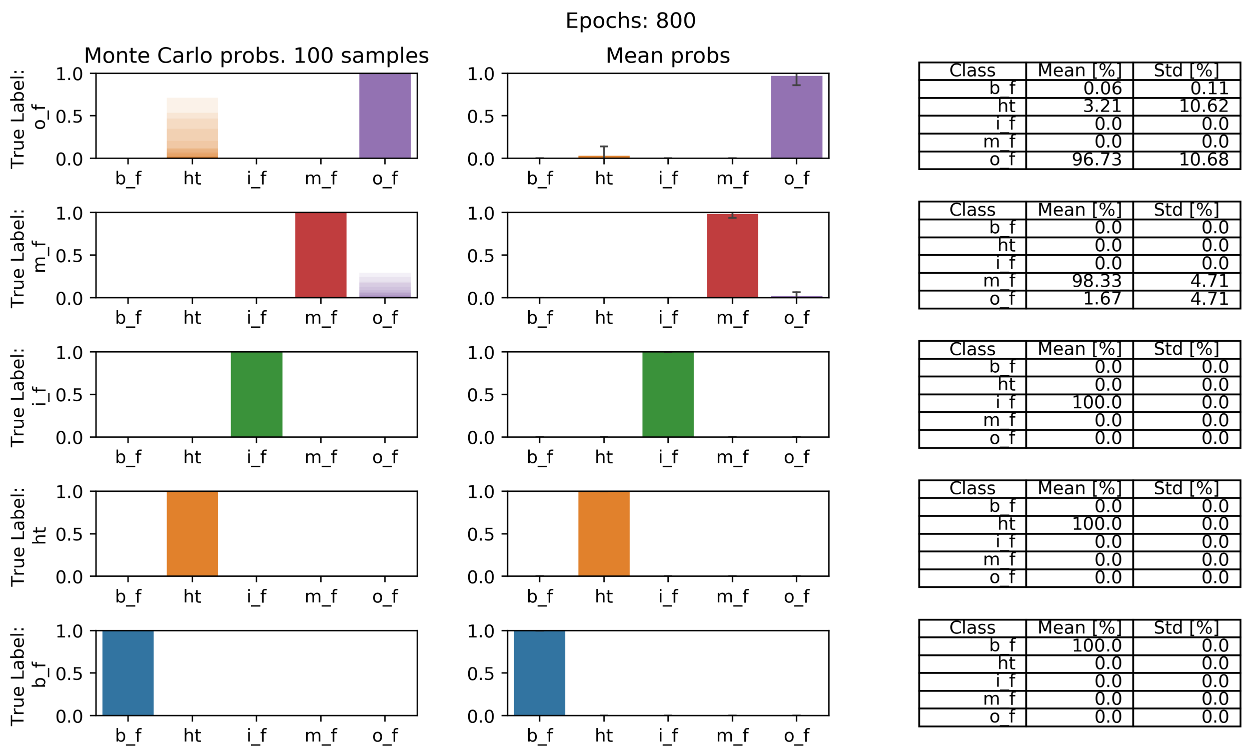


True label \ Predicted label	Healthy	Inner Fault	Outter Fault	Mixed Fault	Ball Fault
Healthy	3102.55 3095.83 3090.35	4.55 2.17 0.00	44.55 39.43 34.45	35.00 32.07 28.00	34.55 30.50 27.00
Inner Fault	6.55 4.23 2.00	3124.55 3118.53 3114.00	6.00 5.07 4.00	66.55 61.53 56.45	13.55 10.63 8.45
Outter Fault	44.55 38.90 32.45	9.00 7.03 5.45	3148.55 3143.10 3137.00	14.55 10.70 7.45	1.00 0.27 0.00
Mixed Fault	49.00 44.77 39.45	45.55 39.97 36.00	30.00 26.70 22.45	3061.00 3053.57 3047.45	40.00 35.00 29.00
Ball Fault	49.10 43.67 37.90	3.00 1.90 1.00	0.55 0.07 0.00	15.55 12.10 9.45	3149.00 3142.27 3136.00

(b)

Figure 5.42: Confusion Matrix 5.42b and normalized confusion matrix 5.42a for the test-set in Ottawa Bearing Dataset using JaNet architecture.

the results obtained with this architecture are outstanding, achieving an accuracy of 97.48 % implies that this model is capable of correctly classify almost all the examples in the test set. As seen in figure 5.40 the model is rarely confused or chooses a different class than the correct one.



Class	Mean [%]	Std [%]
b_f	0.06	0.11
ht	3.21	10.62
i_f	0.0	0.0
m_f	0.0	0.0
o_f	96.73	10.68

Class	Mean [%]	Std [%]
b_f	0.0	0.0
ht	0.0	0.0
i_f	0.0	0.0
m_f	98.33	4.71
o_f	1.67	4.71

Class	Mean [%]	Std [%]
b_f	0.0	0.0
ht	0.0	0.0
i_f	100.0	0.0
m_f	0.0	0.0
o_f	0.0	0.0

Class	Mean [%]	Std [%]
b_f	0.0	0.0
ht	100.0	0.0
i_f	0.0	0.0
m_f	0.0	0.0
o_f	0.0	0.0

Class	Mean [%]	Std [%]
b_f	100.0	0.0
ht	0.0	0.0
i_f	0.0	0.0
m_f	0.0	0.0
o_f	0.0	0.0

Figure 5.43: Distribution for the results in JaNet model for Ottawa Bearing dataset.

The behavior of the loss function illustrated in figure 5.44 is the same as in the case of the VRNN explained in the previous item.

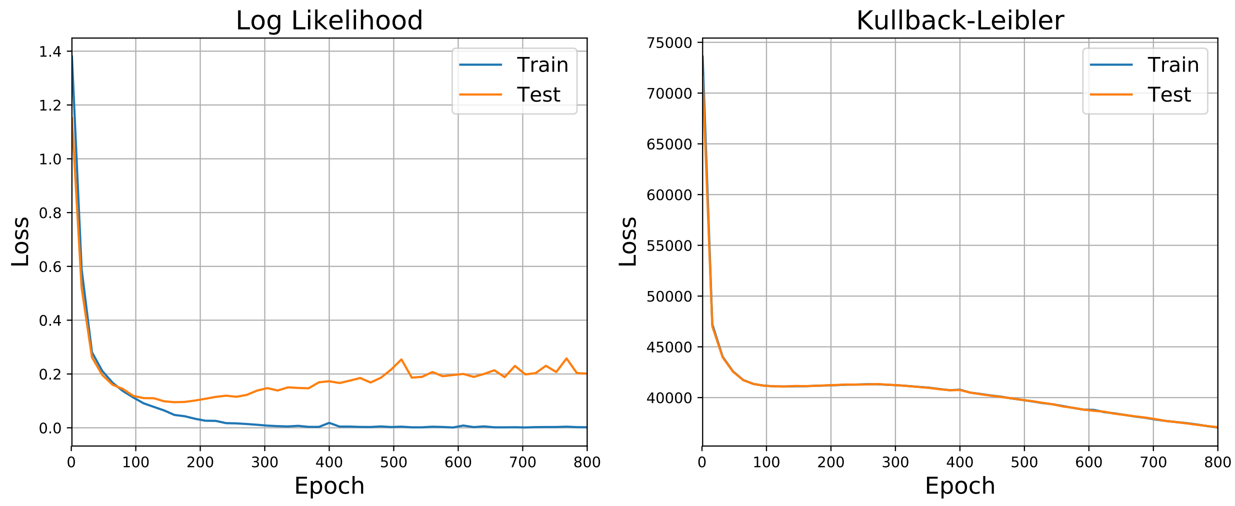


Figure 5.44: Log Likelihood and Kullback-Leibler in Bayes JaNet training for Ottawa Dataset.

As it can be seen in figures 5.39 and 5.42, the results for this classification of data are excellent, the general results for the three test sets and the three iterations of each are presented in the tables below.

Table 5.17: Result for every iteration and dataset for VRNN architecture in Ottawa Bearings Dataset.

VRNN				
Accuracy [%]				
Iteration	1	2	3	Mean
Data_1	90,61	90.65	91,08	91,31
Data_2	96.30	97,30	96.85	96,81
Data_3	93.18	93.65	94.15	93.66

Table 5.18: Result for every iteration and dataset for JaNet architecture in Ottawa Bearings Dataset.

JaNet				
Accuracy [%]				
Iteration	1	2	3	Mean
Data_1	91.06	92.03	91.30	91.46
Data_2	97.40	97.48	97.10	97.32
Data_3	94.45	94.75	94.90	94.70

Finally, as distinguished from the previously illustrated results, the model that is best able to predict the behavior of the data, thus giving a better categorization, is the model based on the JaNet cell, however, both, Bayes JaNet and Bayes VRNN models present excellent results in this dataset. The main advantage of these models compared to the frequentist models, lies in the interpretation of the model, for example, in the case of a frequentist network, when entering a given data into the network, the model will always deliver the same result, being able to be wrong, however, the model developed in this work, being able to quantify its own uncertainty, can show more than one classification, which allows to interpret the values delivered by the network and thus make more pertinent decisions.

5.5. Politecnico di Torino rolling bearing test rig

In this section, rolling bearing test rig is studied in order to diagnose the health state of high speed aeronautical bearings whose accelerometric acquisitions at variable rotational speed, radial load and damage level, together with temperature measurements, are being made available as open access data.

The objective of using this dataset is to use the proposal model for classification in PHM context.

5.5.1. Data Description

The test rig depicted in figure 5.45 consists in high speed spindle, driving the rotation of a shaft. The bearings of the spindle are grease lubricated and their temperature is limited by a liquid (glycol/water) refrigeration circuit, the speed of the spindle is setted by a control system, but the spindle has no keyphasor transducer or tachometer to detect its actual speed and there is no feedback for the controller of the inverter. As consequence, the speed of the shaft is always lower than the ideal one. [8]

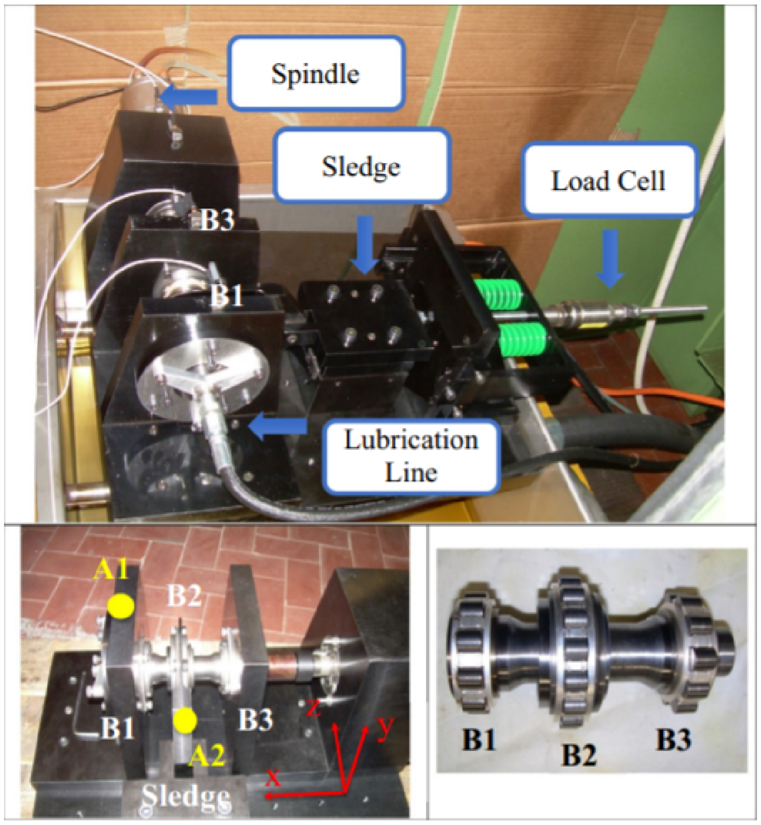


Figure 5.45: The test rig a) general view of the test rig; b) positions of the two accelerometers and the reference system; c) the shaft with its three roller bearings.[8]

The three bearings are put together in the spindle, the outer bearings are identical and the inner rings of the three bearings are connected by a very short and thick hollow shaft designed to speed up to 3600 [rpm].

The main properties of the external bearings (B1 & B3) and the center bearing (B2) are shown below:

Table 5.19: Properties of the bearings in test rig

	Pitch diameter D (mm)	Rollers diameter d (mm)	Contact angle (ϕ)	Rolling elements Z
B1 & B3	40.5	9.0	0	10
B2	54.0	8.0	0	16

Two types of experiments are performed for the dataset, in the first one, the accelerations are relative to the bearings with different damages running at different speeds and under different loads, and the second reports the behavior of a single damaged bearing undergoing a long (about 330 [h]) test at constant speed and load.

The table for the different health conditions of the bearings is shown below:

Table 5.20: Health state on Politecnico di Torino's bearings

Name	Defect	Dimension (μ)
C0A	No defect	–
C1A	Diameter of an indentation on the inner ring	450
C2A	Diameter of an indentation on the inner ring	250
C3A	Diameter of an indentation on the inner ring	150
C4A	Diameter of an indentation on a roller	450
C5A	Diameter of an indentation on a roller	250
C6A	Diameter of an indentation on a roller	150

The testing process of every bearing is illustrated below:

- A brief run at the minimum speed (100 [Hz]) and no load, so to check the correct mounting.
- Application of the static load: at first 1000[N], then 1400 [N] and finally 1800 [N].
- Increment of the speed of the shaft from 0[Hz] to 500 [Hz] with steps of 100 [Hz].
- Measurement of the accelerations as soon as a steady speed of the shaft was reached.

The measurements taken from the experiments are shown below in table 5.21.

Table 5.21: Direction of the measured accelerations

	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
Direction	Axial, X	Axial, Y	Axial, Z	Radial, Z	Radial, Y	Radial, X

5.5.2. Data Preparation

The preprocessing of the data is the same as the one used in subsection 5.4.2 i.e. the first step to operate this data is to obtain feature data from the feature batches, as it is show in figure 5.36, then this feature data is treated as shown in figure 5.37 to obtain the final data for the input of recurrent neural networks, the difference between this processes is that the feature batch size in this case is 300 and the time step is 50.

As the files in the data remain separated through the prepossessing, for test-train division, two files are removed for testing and the remaining files are used for training, this means that 15 % of the dataset is used for testing purposes and 85 % of the dataset is used for training the model.

Before the training-testing process, training data is normalized using a min-max scaler, after the scaler is trained using training data, the test data is normalized using the scaler previously trained.

The training-testing process in this case is running eleven times to ensure that every file is used for testing at least once.

5.5.3. Results and Discussion

The problem relevant to the dataset previously studied corresponds to a classification or diagnosis of the state of health of the bearings, this classification contains seven classes previously explained in the 5.5.2 subsection, therefore, the type of network used for the resolution of this problem is similar to the one in figure 5.38 but instead of having five elements in the output, there are seven.

The number of epochs necessary for the convergence of the solution to this problem is 400, the KL divergence regulator has a value of $1e-4$ and the batch size used is 800. Finally, the number of samples extracted from the model for the generation of the output distributions is 100.

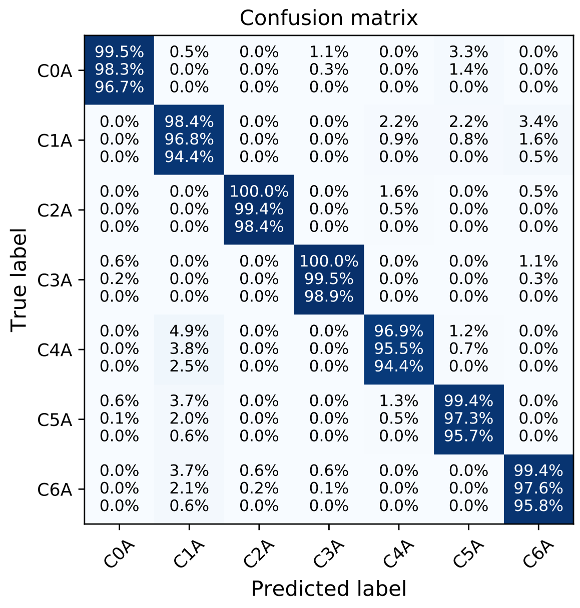
The results obtained for the two architectures studied are presented below.

- Bayes VRNN:
  The optimization process for Bayes RNN using VRNN as recurrent cell is done via grid search, the result of this grid search is shown on table below:

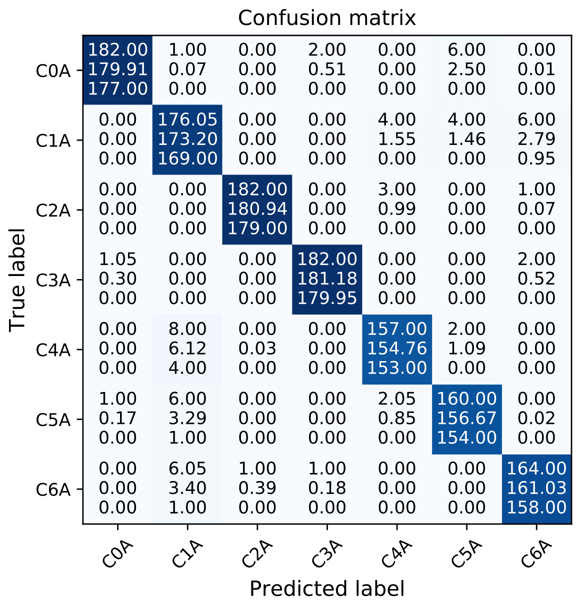
Table 5.22: Parameters for Bayesian VRNN in Torino Bearings Datset.

Layers	Units/Neurons	Activation Function
VRNN	64	Relu
DenseFlipout	64	Relu
DenseFlipout	5	

Using these parameters and the previously explained configurations, the results obtained are illustrated in figure 5.46.



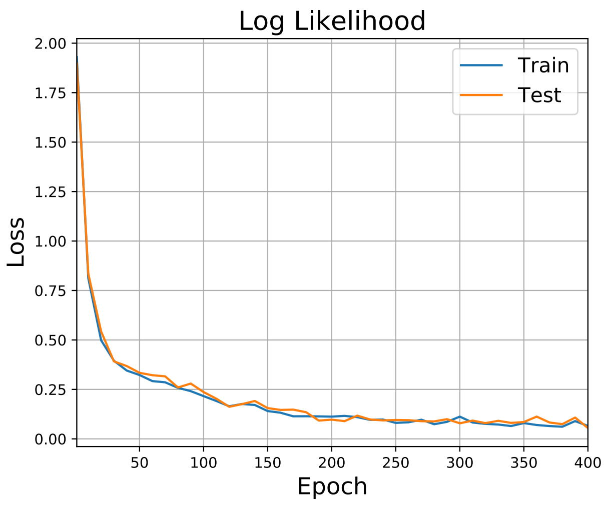
(a)



(b)

Figure 5.46: Confusion Matrix 5.46b and normalized confusion matrix 5.46a for Politecnico di Torino Bearing Dataset using VRNN architecture.

The results for the B-VRNN network conclude with an accuracy of around 91 %, which means that the model used is capable of accurately recognize the characteristics of the examples provided, and using them to predict their condition.



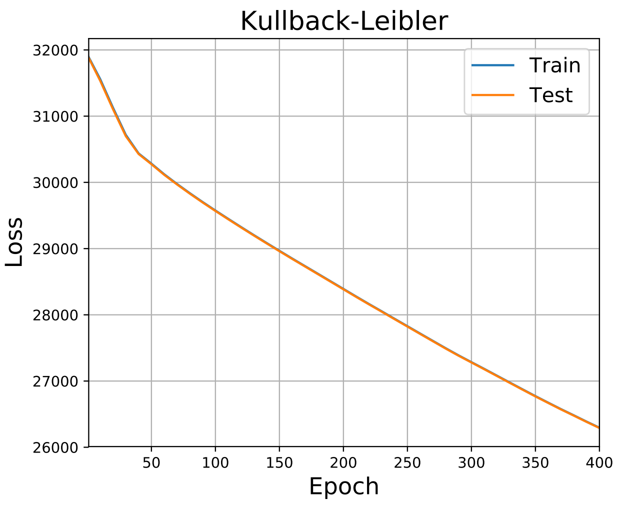


Figure 5.47: Log Likelihood and Kullback-Leibler in Bayes VRNN training for Politecnico di Torino Bearing Dataset.

In this case again both terms of the loss function decrease, as seen in figure 5.47 the term of the log likelihood stabilizes while the term of the KL divergence only decreases, this tells us that despite the proposed distribution ($q(z|\theta)$) gets closer and closer to the prior distribution of weights ($p(z)$) the results do not improve.

- Bayes JaNet:
  The optimization process for this network is the same that in the previous item and the values are similar to those illustrated in table 5.22.

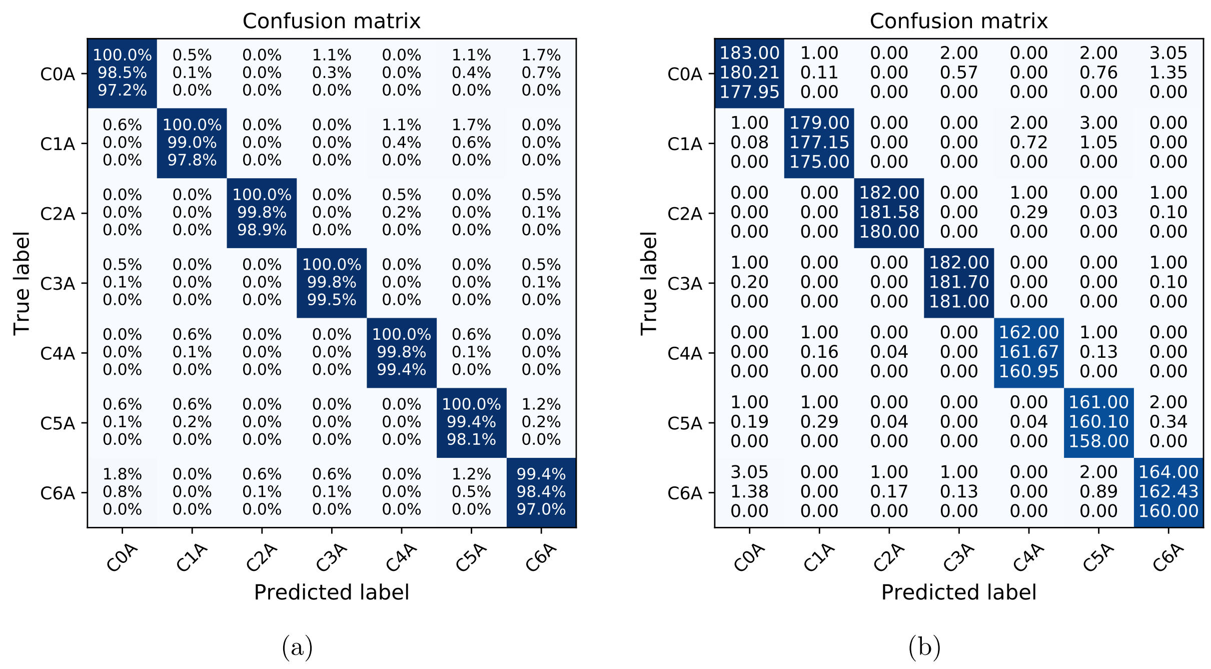


Figure 5.48: Confusion Matrix 5.48b and normalized confusion matrix 5.48a for Politecnico di Torino Bearing Dataset using JaNet architecture.

The studies carried out for this model show that it has a better generalization capacity than the previous model, despite having a similar behavior, this, besides meaning that this architecture has a good generalization capacity for this dataset, it could also signify that the data is not complex enough to represent a challenge for the proposed neural network, however, this serves to demonstrate the efficiency of the algorithm for classification tasks.

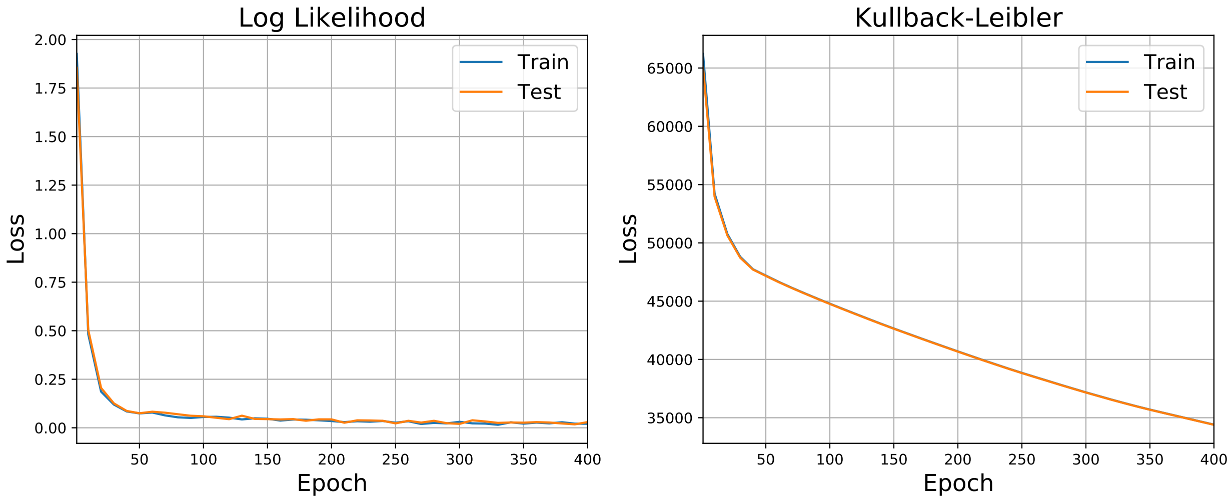


Figure 5.49: Log Likelihood and Kullback-Leibler in Bayes JaNet training for Politecnico di Torino Bearing Dataset.

The behavior of the cost function is similar to the previous case illustrated in the figure

5.47

To visualize the uncertainty in this case, the image below is presented, where it is seen that even when the network is always able to correctly classify the data, in some samples it distinguishes that they could belong to another class.

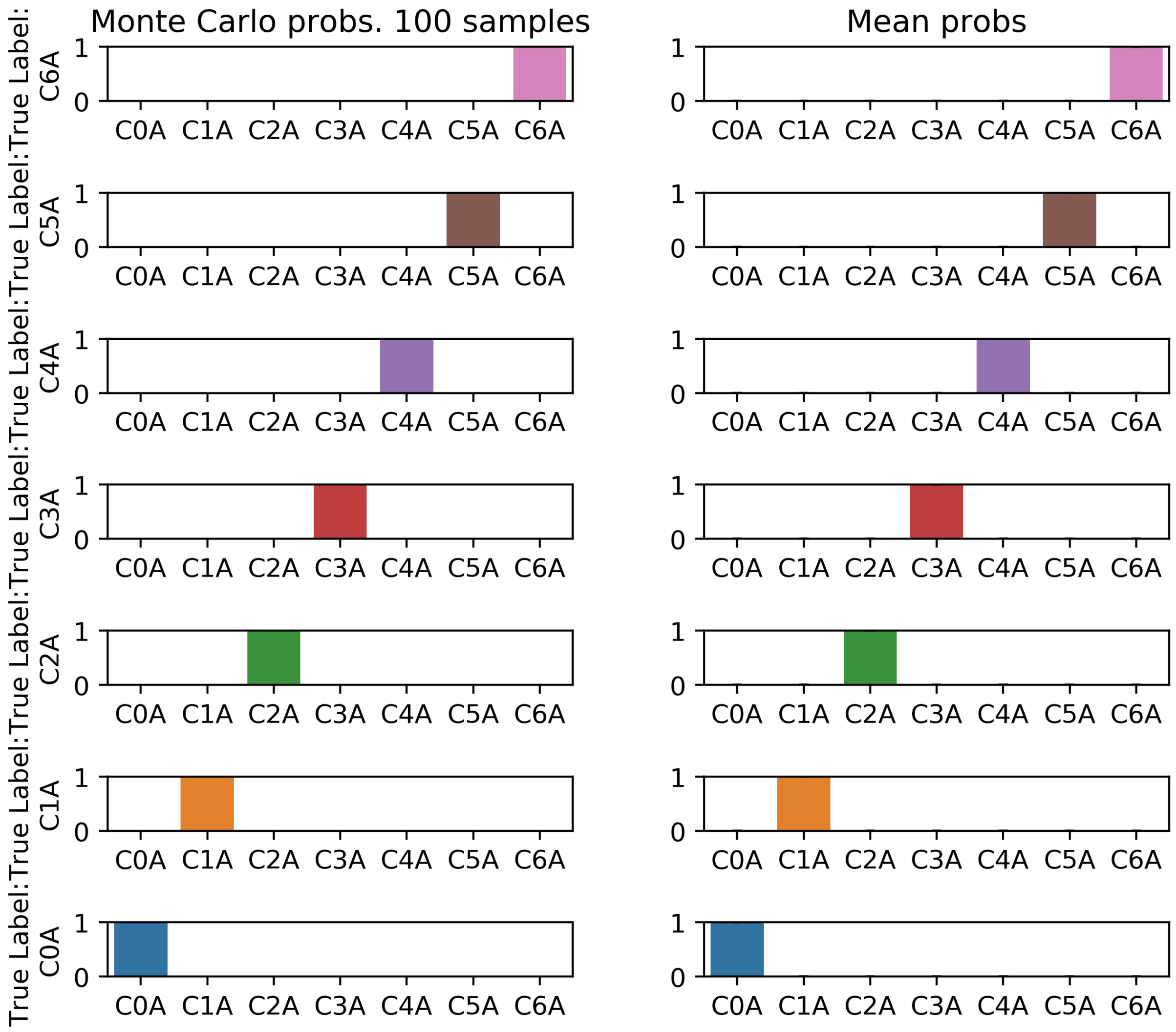
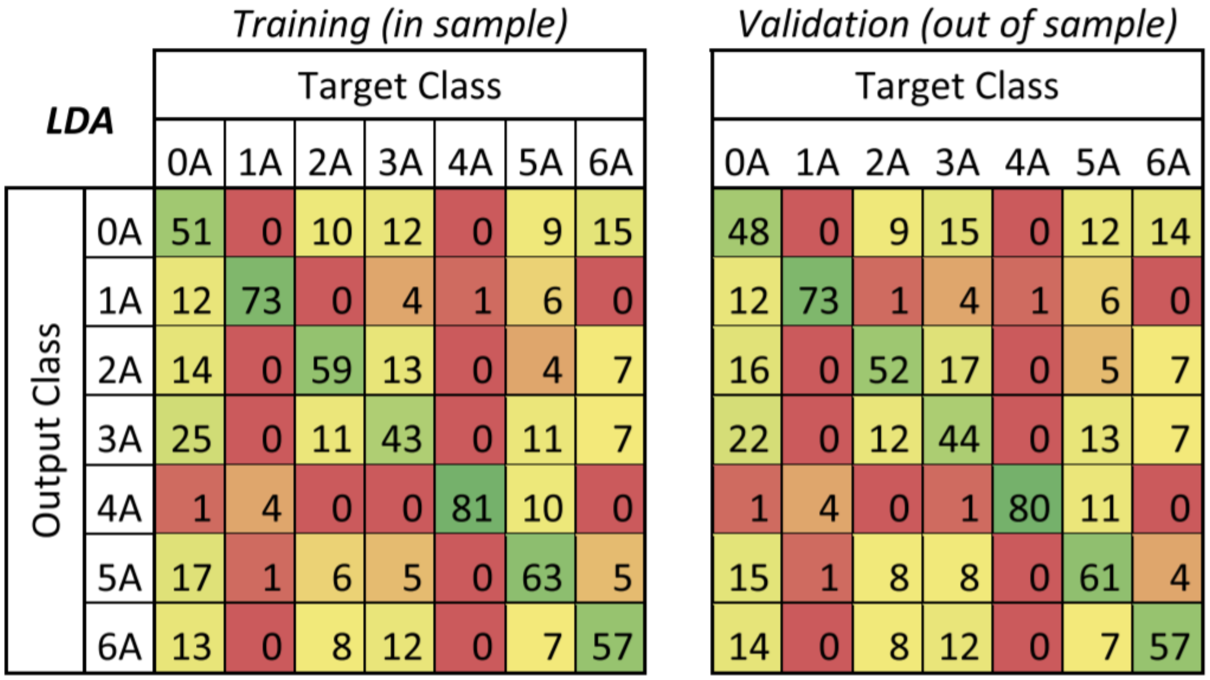


Figure 5.50: Distribution for the results in JaNet model for Politecnico di Torino Bearing dataset.

In the paper that describes the dataset studied above, the confusion matrices illustrated in figure 5.51 are presented, which despite of having worse results than those studied in this paper, the prediction was made with a less complex algorithm (linear discriminant analysis).



Training (in sample)

LDA	Target Class						
Output Class	0A	1A	2A	3A	4A	5A	6A
0A	51	0	10	12	0	9	15
1A	12	73	0	4	1	6	0
2A	14	0	59	13	0	4	7
3A	25	0	11	43	0	11	7
4A	1	4	0	0	81	10	0
5A	17	1	6	5	0	63	5
6A	13	0	8	12	0	7	57

Validation (out of sample)

	Target Class						
Output Class	0A	1A	2A	3A	4A	5A	6A
0A	48	0	9	15	0	12	14
1A	12	73	1	4	1	6	0
2A	16	0	52	17	0	5	7
3A	22	0	12	44	0	13	7
4A	1	4	0	1	80	11	0
5A	15	1	8	8	0	61	4
6A	14	0	8	12	0	7	57

Figure 5.51: LDA confusion matrices in rounded percentages by rows [8].

The most remarkable results in this dataset are the ability of the algorithm developed to predict the health condition of the bearings, together with the ability to determine the uncertainty.

Chapter 6

Concluding Remarks

In this work, Bayesian dense layers are wrapped into Recurrent Neural Networks to propose Bayesian recurrent neural networks for prognosis of remaining useful life (RUL) and health state diagnosis in mechanical equipment. This is a major advantage over the frequentist approach of machine learning since it is possible to quantify uncertainty, which is achieved by outputting a distribution unlike a single number in frequentist networks.

The Bayesian VRNN and JaNet models developed is validated with state of the art performance in the C-MAPSS dataset, where it outperforms other models (MODBNE, DCNN, DLSTM, AdaBN-CNN) in the most difficult sub datasets (FD002 and FD004), which have multiple operational conditions and more examples than the most simple sub datasets (FD001 and FD003), encouraging the scalability with data complexity and dimensionality.

The models are also tested in four different datasets in which it has proven its efficiency to determine the remaining useful life and the health condition of different mechanical equipment. In each of these datasets, the model presents outstanding results, which proves its usefulness not only in simulated data.

In addition to showing highly accurate predictions, the proposed model is able to determine uncertainty, unlike the models previously presented. This ability to determine the uncertainty in the remaining useful life and in the health state of the equipment can be used to make decisions incurring a lower risk of premature failure.

The work presented above, allows to declare that recurrent Bayesian neural networks are a useful and efficient tool in the context of prognostics and health management that allows decisions to be made with a more detailed level of information in order to avoid damage in mechanical equipment.

Bibliography

[1] JULIEA BANNANTINE, JESSJ COMER, and JAMESL HANDROCK. Fundamentals of metal fatigue analysis((book)). *Research supported by the University of Illinois. Englewood Cliffs, NJ, Prentice Hall, 1990, 286*, 1990.

[2] Eric Bechhoefer, Brandon Van Hecke, and David He. Processing for improved spectral analysis. In *Annual conference of the prognostics and health management society*, pages 14–17, 2013.

[3] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Camila Correa-Jullian, José Miguel Cardemil, Enrique López Droguett, and Masoud Behzad. Assessment of deep learning techniques for prognosis of solar thermal systems. *Renewable Energy*, 145:2178–2191, 2020.

[8] Alessandro Paolo Daga, Alessandro Fasana, Stefano Marchesiello, and Luigi Garibaldi. The politecnico di torino rolling bearing test rig: Description and analysis of open access data. *Mechanical Systems and Signal Processing*, 120:252–273, 2019.

[9] Márcio das Chagas Moura, Enrico Zio, Isis Didier Lins, and Enrique Droguett. Failure and reliability prediction by support vector machines regression of time series data. *Reliability Engineering & System Safety*, 96(11):1527–1534, 2011.

[10] L Deng, D Yu, et al. Deep learning: methods and applications. found trends signal process 7 (3–4): 197–387, 2014.

[11] ASTM Designation. E466-07, standard practice for conducting force controlled constant amplitude axial fatigue tests of metallic materials. *ASTM International*, 2007.

[12] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

[13] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural net-

works. *arXiv preprint arXiv:1704.02798*, 2017.

[14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[15] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, 2019.

[16] Kai Goebel, Bhaskar Saha, Abhinav Saxena, Jose R Celaya, and Jon P Christophersen. Prognostics in battery health management. *IEEE instrumentation & measurement magazine*, 11(4):33–40, 2008.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[18] Huan Huang and Natalie Baddour. Bearing vibration data collected under time-varying rotational speed conditions. *Data in brief*, 21:1745–1749, 2018.

[19] Feng Jia, Yaguo Lei, Jing Lin, Xin Zhou, and Na Lu. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, 72:303–315, 2016.

[20] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[21] Samir Khan and Takehisa Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical Systems and Signal Processing*, 104:799–834, 2018.

[24] Jialin Li, Xueyi Li, and David He. Domain adaptation remaining useful life prediction method based on adabn-dcnn. In *2019 Prognostics and System Health Management Conference (PHM-Qingdao)*, pages 1–6. IEEE, 2019.

[25] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172: 1–11, 2018.

[26] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33–47, 2018.

[27] M McDade and K Munch. Gearbox reliability collaborative: gearbox inspection metadata. *NREL/TP-500-49133*, 2010.

[28] Patrick L McDermott and Christopher K Wikle. Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. *Entropy*, 21(2):184, 2019.

[29] Huixing Meng and Yan-Fu Li. A review on prognostics and health management (phm)

methods of lithium-ion batteries. *Renewable and Sustainable Energy Reviews*, 116: 109405, 2019.

[30] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, volume 192, 2016.

[31] Art B Owen. Monte carlo theory, methods and examples. *Monte Carlo Theory, Methods and Examples. Art Owen*, 2013.

[32] Tim Pearce, Nicolas Anastassacos, Mohamed Zaki, and Andy Neely. Bayesian inference with anchored ensembles of neural networks, and application to exploration in reinforcement learning. *arXiv preprint arXiv:1805.11324*, 2018.

[33] Weiwen Peng, Zhi-Sheng Ye, and Nan Chen. Bayesian deep-learning-based health prognostics toward prognostics uncertainty. *IEEE Transactions on Industrial Electronics*, 67 (3):2283–2293, 2019.

[34] Ingo Rechenberg. Evolutionsstrategie—optimierung technischer systeme nach prinzipien der biologischen information. *Stuttgart-Bad Cannstatt: Friedrich Frommann Verlag*, 1973.

[35] Stuart Russel, Peter Norvig, et al. *Artificial intelligence: a modern approach.* Pearson Education Limited, 2013.

[36] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

[37] David Siegel, Hassan Al-Atat, Vishwesh Shauche, Linxia Liao, John Snyder, and Jay Lee. Novel method for rolling element bearing health assessment—a tachometer-less synchronously averaged envelope feature extraction technique. *Mechanical Systems and Signal Processing*, 29:362–376, 2012.

[38] T Touret, C Changenet, F Ville, M Lalmi, and S Becquerelle. On the use of temperature for online condition monitoring of geared systems–a review. *Mechanical Systems and Signal Processing*, 101:197–210, 2018.

[39] Jos Van Der Westhuizen and Joan Lasenby. The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*, 2018.

[40] Brandon Van Hecke, Yongzhi Qu, David He, and Eric Bechhoefer. A new spectral average-based bearing fault diagnostic approach. *Journal of Failure Analysis and Prevention*, 14(3):354–362, 2014.

[41] David Benjamin Verstraete, Enrique López Droguett, Viviana Meruane, Mohammad Modarres, and Andrés Ferrada. Deep semi-supervised generative adversarial fault diagnostics of rolling element bearings. *Structural Health Monitoring*, page 1475921719850576, 2019.

[42] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066, 2013.

[43] Dong Wang and Kwok-Leung Tsui. Statistical modeling of bearing degradation signals. *IEEE Transactions on Reliability*, 66(4):1331–1344, 2017.

[44] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[45] Jun Wu, Kui Hu, Yiwei Cheng, Haiping Zhu, Xinyu Shao, and Yuanhang Wang. Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network. *ISA transactions*, 97:241–250, 2020.

[46] Huisung Yun. *Entropic approaches for assessment of metal fatigue damage*. PhD thesis, 2019.

[47] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10):2306–2318, 2016.

[48] Yongzhi Zhang, Rui Xiong, Hongwen He, and Michael G Pecht. 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, 2018.

[49] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.