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

BAYESIAN VARIATIONAL RECURRENT NEURAL NETWORKS FOR PROGNOSTICS AND HEALTH MANAGEMENT OF COMPLEX SYSTEMS.

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

DANILO FABIÁN GONZÁLEZ TOLEDO

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
RODRIGO PASCUAL JIMÉNEZ

Este trabajo ha sido parcialmente financiado por Beca Magíster Nacional CONICYT

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: DANILO FABIÁN GONZÁLEZ TOLEDO
AÑO: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

BAYESIAN VARIATIONAL RECURRENT NEURAL NETWORKS FOR PROGNOSTICS AND HEALTH MANAGEMENT OF COMPLEX SYSTEMS.

In recent couple of years many automated data analytics models are implemented, they provide solutions to problems from detection and identification of faces to the translation between different languages. This tasks are humanly manageable but with a low processing rate than the complex models. However, the fact that are humanly solvable allows the user to agree or discard the provided solution. As an example, the translation task could easily output misleading solutions if the context is not correctly provided and then the user could improve the input to the model or just discard the translation.

Unfortunately, when this models are applied to large databases is not possible to apply this 'double validation' and the user might believe blindly in the model output. The above could lead into a biased decision making, threatening not only the productivity but also the security of the workers.

Because of this, is neccessary to create models that quantify the uncertainty in the output. At the following thesis work the possibility to use distributions over single point matrices as weights is fused with recurrent neural networks (RNNs) a type of neural network which are specialized dealing with sequential data. The model proposed could be trained as discriminative probabilistic models thanks to the Bayes theorem and the variational inference.

The proposed model is called 'Bayesian Variational Recurrent Neural Networks' is validated with the benchmark dataset C-MAPSS which is for remaining useful life (RUL) prognosis. Also, the model is compared with the same architecture but a frequentist approach (single points matrices as weights), with different models from the state of the art and finally, with MC Dropout, another method to quantify the uncertainty in neuronal networks. The proposed model outperforms every comparison and furthermore, it is tested with two classification tasks in bearings from University of Ottawa and Politecnico di Torino, and two health indicator regression tasks, one from a commercial wind turbine from Green Power Monitor and the last in fatigue crack testing from the University of Maryland showing low error and good performance in all tasks.

The above proves that the model could be used not only in regression tasks but also in classification. Finally, it is important to notice that even if the validations are in a mechanical engineering context, the layers are not limited to them, allowing to be used in another context with sequential data.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: DANILO FABIÁN GONZÁLEZ TOLEDO
AÑO: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

BAYESIAN VARIATIONAL RECURRENT NEURAL NETWORKS FOR PROGNOSTICS AND HEALTH MANAGEMENT OF COMPLEX SYSTEMS.

En el último par de años se han implementado muchas soluciones basadas en el análisis automático de datos, desde la detección e identificación de rostros a la traducción de diferentes idiomas. Éstas tareas son humanamente realizables pero con una tasa de procesamiento mucho menor que la obtenida gracias a los algoritmos de aprendizaje de máquinas. Sin embargo, el que sean humanamente realizables permite decidir si alguno de los resultados entregados por el modelo es confiable o no. Un ejemplo de lo anterior puede ser la traducción de un idioma a otro, donde muchas veces características como el contexto son importantes y no son directamente introducidas al traductor.

Lamentablemente, al utilizar estos modelos de aprendizaje automático con largas bases de datos sensoriales no es posible realizar esta 'doble validación' y se suele creer ciegamente en el resultado del modelo. Lo anterior puede originar una toma de decisiones sesgada poniendo en peligro no sólo la productividad sino que también la seguridad de los operadores.

Debido a esto, es necesario originar modelos que puedan cuantificar la incertidumbre en su salida. En la tesis a continuación se utiliza la posibilidad de emplear distribuciones en vez de matrices de pesos puntuales con las redes neuronales recurrentes (RNNs), un tipo de red neuronal especializada para trabajar con datos secuenciales. El modelo propuesto puede ser entrenado como un modelos probabilístico discriminativo gracias a la aplicación del teorema de Bayes y la inferencia variacional.

Este modelo propuesto, llamado 'Bayesian Variational Recurrent Neural Networks' es validado con la base de datos C-MAPSS para regresión de la vida útil remanente, en donde se compara con la misma arquitectura en su versión frecuentista, con diferentes modelos del estado del arte y finalmente con MC Dropout, otro método utilizado para evaluar la incertidumbre en redes neuronales. El modelo propuesto obtiene mejores resultados en todas las comparaciones anteriores y también se demuestra su utilización en dos problemas de clasificación de estado de salud con bases de datos de rodamientos provenientes de la Universidad de Ottawa y el Politecnico di Torino y regresión de indicador de vida para una turbina eólica comercial de Green Power Monitor y una base de datos de grietas en un ensayo de fatiga de la Universidad de Maryland con bajo porcentaje de error y buen desempeño.

Lo anterior demuestra que el modelo propuesto puede ser utilizado no sólo en regresión, sino también en clasificación. Finalmente es importante destacar que se valida pero no se restringe su utilización en otros contextos con bases de datos secuenciales.

Hoy se te da,
hoy se te quita.

Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Introduction

In general, as new technologies are developed, the complexity of the components often grows. This complexity consider the specialization to do a task, their interaction with other components and their importance in the organization, among others. In a industrial context this could be worrying, since the failure in some of this components affects directly the expected results and, commonly, due to their specialization, the reparations are expensive (in time and cost).

Therefore, it is necessary to know their failure modes, the physics that rules the functioning states, etc. However, this is a difficult task that requires a lot of knowledge and investigation. Also, it is noticeable that different operation conditions could lead into completely different failures.

Prognostics and health management (PHM) is the specialized area of the engineering that is focused on modelling the lifecycle of components and systems to predict when the prior will no longer perform as intended. The above could be achieved by two main approaches: model-based prognostics and data-driven prognostics. The first ones uses equations that incorporate physical understanding of the system which needs a lot of experimental studies and works under specified conditions, an example for this models is the Paris law, which models the crack growth in a fatigue stress regime.

On the other hand, data-driven models take advantage of the advances of 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. Some useful tools as the Internet of Things (IoT) which helps to monitor and control the operation of the components by multiple network-connected sensors could be the source of the data implying a nice connectivity between data and the application. However, this creates such big databases that human based monitoring is non-viable. One approach to solve this problem is to preprocess the databases to decrease the dimensionality and be allowed to indentify in the data the key features. Data-driven models like machine learning techniques can be used to extract useful information from the data and mix them to understand insights of the system performance [3].

Most recent PHM studies focus on the critical machinery components including bearings [4, 5], gears [6, 7] and batteries [8, 9]. 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 and extract features.

To complete the end-to-end process, it is necessary to use an algorithm that could handle the large amount of big machinery data by learning different features and interactions between them. Here is where, Deep learning has emerged as a promising solution since the

layer to layer architecture enable to extract more abstract and complex information (even features that misses a direct physical meaning) that are useful to perform a certain task. 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. Since this models could be used as components (layers) of a complex model, they are new architectures that mix many of the above as in 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 [10]. Verstraete et al. developed an unsupervised and a semi-supervised deep learning enabled generative adversarial network-based methodology for fault diagnostics of rolling element bearings [11]. Zhang et al. presented a LSTM RNN model to predict the remaining useful life of lithium-ion batteries [12]. Correa-Jullian et al. [13] demonstrated that the deep learning models for solar thermal systems can outperform naïve 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. [14, 15].

It is important to highlight that all the previous studies did not address the uncertainty in the data and/or in the model. This is because their weights are single point estimates and since the operations between layers corresponds to matrix multiplications, for a given input and trained model, the output will be always the same. This may provide overly confident predictions and results, causing poor guidance to the decision-maker team which could lead into safety hazards, lower availability and higher costs. To overcome this problem, Peng et al. [16] proposed a Bayesian deep learning approach to address uncertainty through health prognosis. However, their Bayesian approximation is implemented through Monte Carlo Dropout (MC dropout) that is not truly Bayesian [17]. Unlike the regular dropout, MC Dropout is applied at both training and test, resulting into randomly ignored parts of the network at the testing and with this, different outputs. This ultimately generates random predictions as samples from a probability distribution which is treated as variational inference [18]. However, the performance of MC Dropout has been criticized by Osband et al. [19, 20], and counterexamples were presented to demonstrate the limitations of MC Dropout to produce satisfactory results.

Nowadays, Bayesian recurrent neural networks have been studied and developed in different ways: [21] using a Markov Chain Monte Carlo variation (PX-MCMC) or [22] using Bayes By Backprop with reparameterization trick to estimate the uncertainty. Finally, Wen et al. [23], proposses the flipout method which outperforms previous methods using Bayesian networks.

In the following Master Thesis the Bayesian Neural Networks are developed with the flipout estimator to address the prediction of remaining useful life, classification and health indicator for different applications of datasets with time dependant features (i.e. sensory data). Because of this, the implementation is focused in RNN cells (LSTM and GRU) due to their known performance with sequential data.

The proposed approach is validated using an open-access turbofan engine dataset [24] that is the most usual benchmark for RUL prognosis (C-MAPSS) and also tested with two health indicator prognosis datasets (GPM wind turbine and Maryland cracklines dataset)

and two classification applications (Politecnico di Torino rolling bearing data and Ottawa bearing vibration data).

1.2 Motivation

As summarized above, Bayesian Recurrent Neural Networks (B-RNN) are recurrent neural networks that their weights are defined by a sample from a trained distribution rather than single point estimates.

With this, is possible to draw a distribution in the output of the network which can be sampled to get the final value (class probabilities for classification, remaining useful life or health indicator for regression).

The above distributions helps to generate confidence intervals which includes uncertainty assessment which is relevant in the context of Prognostic health management (PHM) because this confidence intervals help to identify the risk involved in maintenance decisions. On the other hand, single point estimates lacks of this quantification and due to this, their prognostics have no uncertainty in its values, this could be a problem in maintenance tasks because if a mechanical component change its behavior due to different operational conditions this is not represented in the approach, but a larger distribution range notifies it.

With the above, it is substantive to do some robust predictions with rather more useful information than a single value.

1.3 Aim of the work

1.3.1 General objective

Propose Bayesian Recurrent Neural Network models that perform Bayesian variational inference with sequential data with its own framework. Including the develop of the models and their validation of this scheme with classification and regression for Prognostics and Health Management (PHM) of complex systems.

1.3.2 Specific objectives

The specific objectives consists in:

1. Transform the frequentist approach of Recurrent neural networks to a Bayesian scheme.
2. Implement the above cells with Tensorflow Probability dense layers as the gates of the "Long-Short Term Memory" and "Gated Recurrent Units" in a keras-like implementation.
3. Compare benchmark dataset (C-MAPSS) performance with the frequentist (non Bayesian) approach for benchmark dataset.
4. Compare benchmark dataset their performance with Montecarlo Dropout as Bayesian interpretation in a Neural Network for benchmark dataset.
5. Compare the proposed approach performance with the state of the art results for the benchmark dataset.

6. Test other case studies in PHM context as Remaining useful life prognosis and Health indicator prognosis (GPM wind turbine dataset and Maryland crack lines dataset) and health state identification (Ottawa bearing dataset and Politecnico di Torino bearing dataset).

1.4 Resources available for this Thesis

To develop and test this thesis, a desktop computer were provided by Smart Reliability and Maintanance Integration (SRMI) Laboratory of the University of Chile with the following specifications:

- Windows 10 Pro.
- Intel Core i7-8700 CPU 3.20Ghz.
- 32Gb RAM.
- NVIDIA GeForce RTX 2080 Ti.

Also, the software and libraries needed, includes:

- Python 3.6 as programing language.
- TensorFlow 1.13 as deep learning framework.
- TensorFlow Probability 0.6 to combine probabilistic models and deep learning library.
- Numpy, Pandas and SkLearn to sort and preprocess the databases.
- Matplotlib to visualize the database and their results.
- Any dependence of the above libraries.

Finally, it is important to remark that financial support were provided by CONICYT during the full duration of the Master Studies and Thesis through a scholarship.

1.5 Structure of the work

First, in Chapter 2 the methodology of the work is show. Then, in Chapter 3 the Theoretical Background summarizes all the prior knowledge needed to understand the proposed approach including a brief explanation of machine learning, the tasks that are commonly solved with this technique, the most used layers in deep learning (i.e. dense, convolutional and recurrent), Bayesian inference and weight perturbations, which are explained in Chapter 4. After that, in Chapter 5 the different study cases are presented in different sections for regression and classification. Each dataset have a data description, pre-processing and final architecture. Later, in Chapter 6 the results are presented, with a graphic presentation and a discussion about the highlights. Finally the conclusion of all the experiments to summarize the performance and proyection of the Bayesian Recurrent Neural Network.

Chapter 2

Methodology

This thesis investigation explore Bayesian Neural Networks with Flipout as weight perturbation method. To address this research, literature review, development and validation steps are needed, which are explained below:

1. **Literature review**: This consider the exploration of the state of the art in Bayesian Neural Networks, different approaches to Bayesian Recurrent Neural Networks and their applications in Prognostics and health management (PHM).

   Also, the above requires elaborated knowledge in neural networks, Variational inference, backpropagation, backpropagation through time, bayes by backprop and weight perturbations.
2. **Development of Bayesian Recurrent Neural Networks**: This methodological step consists in the ensemble of Bayesian Dense Layers into Recurrent Neural Networks cells joining the acquired knowledge in the prior methodological step.
3. **Dataset selection**: Since Deep learning models are a data-driven technique, it is necessary to look for datasets that presents a regression or classification tasks. Also, since this thesis works with recurrent neural networks, the datasets has to be time series. The datasets are divided in simulated data and real data.
   - **Simulated dataset**: First, since this is a new approach, it is necessary to benchmark the results with a common dataset used in PHM. This is achieved with C-MAPSS dataset, which is a simulated dataset from NASA of a turbofan engine and is widely used in RUL address.
   - **Experimental Dataset**: Since the deal with PHM algorithms is to implement them in real applications, many datasets are included. 3 university datasets including Maryland (Crack lines), Ottawa (Bearing vibration data) and Politecnico di Torino (Rolling bearing data). Finally a real scenario is included: GPM systems (Wind turbine data).
4. **Hyperparameter tuning with the datasets**: In this methodological step a proper B-RNN is designed and optimized for each application. This consider a grid search of the hyperparameters of the network in pursuit of the best results.
5. **Analysis of the results**: After all the above, it lasts the comparison of the proposed model with the current article research. Since C-MAPSS is the benchmark dataset, the results obtained in this dataset are compared with other kind of networks (state of the art) and with another type of uncertainty estimator in neural networks (MC Dropout). Then, the results for the other dataset are presented with their respective metrics and plots. Which leads to the concluding remarks of this thesis.

Chapter 3
Theoretical Background

The following chapter summarizes the necessary concepts to understand the developed framework. Firstly, machine learning has to be defined and reviewed, integrating the most common problems that are solved with it. Then, deep learning expands the above definitions, including neural networks as the vanilla layer and some more specialized ones, like a brief summary of convolutional neural networks (since this type of layer is used but is not the focus of this thesis) and recurrent neural networks with emphasis in two types of cell including Long-Short Term Memory and Gated Recurrent Unit. Finally, a section about Bayesian Inference and Bayes by Backprop as the needed Bayesian background.

3.1 Machine Learning

Machine Learning is a subfield of Artificial Intelligence (AI) whose motivation is to generate inteligent programmation, this means, that the tasks that solve are not directly programmed in the code [25].

Some of the most known applications of Machine Learning are: face recognition (e.g. Facebook [26]), speech recognition (e.g. Alexa, Cortana or Google home [27]) and self-driving cars (e.g. Tesla [28]).

In simple words, the pipeline of a machine learning algorithm consists in a training stage, where the data is provided to the network generating "knowledge". Then, after multiple iterations of this learning process, a model is ready to be used to predict. The prediction stage is when the model is fed with new (unviewed) data and retrieve an output based onto the prior knowledge.

It is possible to classify the tasks that a machine learning algorithm solve into four families: clustering, anomaly detection, classification and regression. Also, the training of the algorithm could be in three ways: supervised, unsupervised and reinforcement learning. This thesis consider classification and regression tasks trained with supervised data.

3.1.1 Supervised learning

At the training time of a machine learning algorithm, the network is trained using the data and the expected result (labels) to quantify the difference between the prediction made by the computer and the ground truth. Since this metric is representative of how well is performing the algorithm, minimizing this function is the goal of any supervised machine learning algorithm.

Due to the above, any supervised learning algorithms generate a map from the database to a label and the loss function changes depending on the data and the task to solve.

3.1.2 Regression and classification tasks

Classification is a task where the computer must decide in which of n categories some input belongs to. This is commonly solved by a function $f : \mathbb{R}^k \to \{1, ..., n\}$ learned by the algorithm [10]. In most cases, the output is interpreted as the probability of being part of a class and the function that represents a categorical probability function is the SoftMax:

$$\sigma(z) = \frac{\mathrm{e}^{z_j}}{\sum_{\mathrm{i}=1}^{n} \mathrm{e}^{z_\mathrm{i}}} \qquad for\, j \in \{1, ..., n\} \tag{3.1}$$

In the regression task, the above function changes to $f : \mathbb{R}^k \to \mathbb{R}$, meaning that the function maps all the inputs to a single value and its most known application is to predict or forecast some behavior driven by sequential data [10]. Here, the most used loss function is the root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{\mathrm{i}=1}^{n}(Y_\mathrm{i} - \hat{Y}_\mathrm{i})^2} \tag{3.2}$$

3.2 Deep Learning

Deep Learning is an specialized type of machine learning that create internal abstractions of the feeded data by non-linear transformations made by "layers". This layers could be artificial neural networks (dense layers), convolutional neural networks or recurrent neural networks. Also, they are layers for dimensionality reduction (pooling layers) and for regularization (dropout).

The word "deep" is used because the layers could be stacked, with this is possible to extract more complex features. Also, the features extracted are the most useful to achieve the goal since they are optimized by minimizing a loss function that is created for a particular task. The above, eliminates the prior feature engineering common to all statistic methods, allowing to model complex systems and solve problems without further physical knowledge [**?**].

3.2.1 Artificial Neural Networks

An artificial neural networks (ANN) is a connection between nodes known as artificial neurons. They are inspired by biological neural networks since emulates the chemical connection in the biological neurons with a computation of a non linear function of its inputs. The non linear function is created by a linear combination and a non linear activation function. The parameters of the linear combination are called weights and are trained by the network.

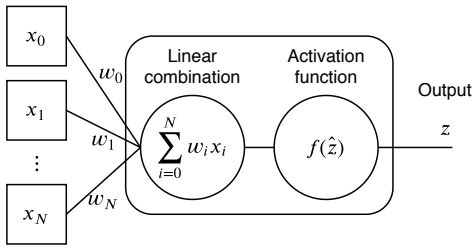


Figure 3.1: Neuron.

A stack of neurons conforms a layer known as dense layers because every neuron interacts with all input features and a stack of layers generate a multilayer perceptron (MLP) or artificial neural network (ANN). With this, the number of parameters increases with the multiple connections between layers.

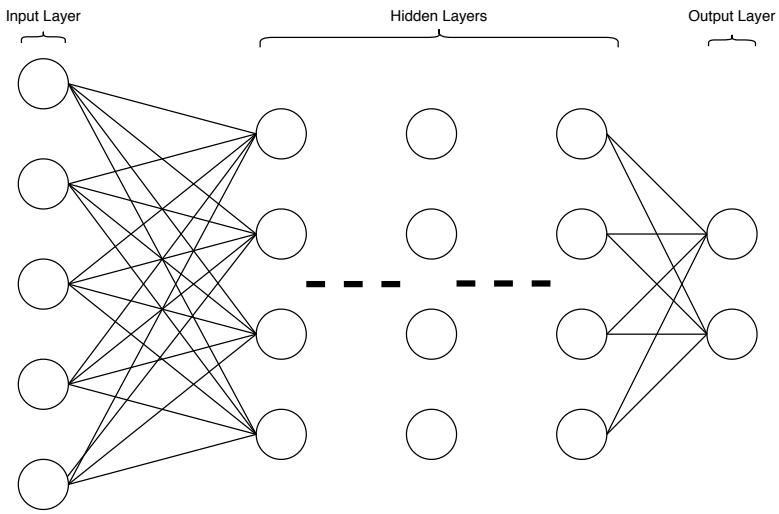


Figure 3.2: Aritificial neural network.

3.2.2 Convolutional neural networks

Is noticeable that dense neural networks makes sense with one dimensional (or flattened) data. For two dimensional (or grid) data, as images, the local relationship between nearby pixels is an important feature and dense layers could not learn this kind of link. To deal with this, Convolutional neural network uses small kernels of learnable weights that are slided through the data with a mathematical function called convolution:

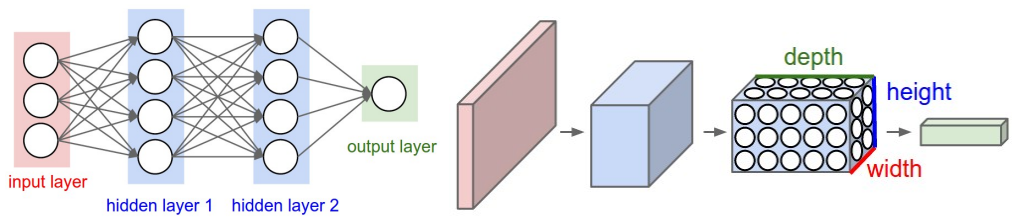
$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{3.3}$$

Where $s(t)$ is the output for the kernels $w(t)$ convolved onto the input $x(t)$. The above is with one dimentional data, but is easily constructed for 2 dimentional data:

$$S(\mathrm{i}, j) = (K * I)(\mathrm{i}, j) = \sum_m \sum_n I(\mathrm{i} + m, j + n) K(m, n) \tag{3.4}$$

With all the above, it follows that Convolutional neural networks have three [10] main capabilities:

- **Sparse interactions**: Since the kernels are noticeably smaller than the input, the number of trainable parameters is low in comparison to dense layers. This also implies fewer operations improving the efficiency of the algorithm.
- **Shared weights**: Also, this kernels are slided through the input, therefore the weights are used multiple times in the model. In a dense layer, the weights are used only once since they are associated with only one feature of the input.
- **Equivariance to translation**: In the convolution operation is possible to apply a translation function and then a convolution or the convolution and then the translation and the transformation of the input is the same. This means that the extraction of the parameters is sequential so the representation is a map of the represented features no matter where they are.



(a) Dense neural network (b) Convolutional neural network

Figure 3.3: Graphical comparison between Dense and Convolutional layers. Source: `http://cs231n.stanford.edu/`

The Figure 3.3 shows the differences between dense layers and convolutional layers, the most important one is that convolutions generate smaller images with more features as outputs thanks to the smaller kernels and dense layers output as many weights.

3.2.3 Recurrent neural networks

The main topic of this thesis are Recurrent Neural Networks (RNN), therefore, it is important to know the most important features from this type of neural network.

First of all, as convolutional neural networks takes advantage of their small kernels to extract useful features in grid data, recurrent neural networks are specialized for processing sequential data.

To achieve this, the possibility to share weights plays an important role since this cause a generalization across all the inputs that are operated with the same weights. This is why RNNs share weights across time and defines that the function created by the network maps the state at time t to time $t + 1$ (with the same parameters). Mathematically, RNNs could be described as dynamical system, defined as follows:

$$s^{(t)} = f(s^{(t-1)}; \theta) \tag{3.5}$$

Where $s(t)$ is the state of the system and θ the parameters (weights).

At the above definition is possible to observe that the time t is defined by the previous time $t-1$ and then it is possible to "unfold" this recurrent graph, for example with $t = 4$:

$$\begin{aligned} s^{(4)} &= f(s^{(3)}; \theta) \\ &= f(f(s^{(2)}; \theta); \theta) \end{aligned} \tag{3.6}$$

If we continue unfolding this equation, a non recurrence expression is generated. Also, this state $s(t)$ is known as hidden state $h(s)$, as is an internal process of the recurrent network.

Then, is noticeable that the above expression does not depend on the input, because the dynamic system that we model is only dependant at the prior state. Now, is possible to incorporate the input $x(t)$, doing that, the hidden state depends at the prior hidden state and the input at time t:

$$h^{(t)} = f(h^{(t-1)}, x(t); \theta) \tag{3.7}$$

With this, the state now contains information about the current input and the last hidden state, being that, the extracted features of the whole sequence.

This new expression can be represented as a folded and a directed graph as follows:

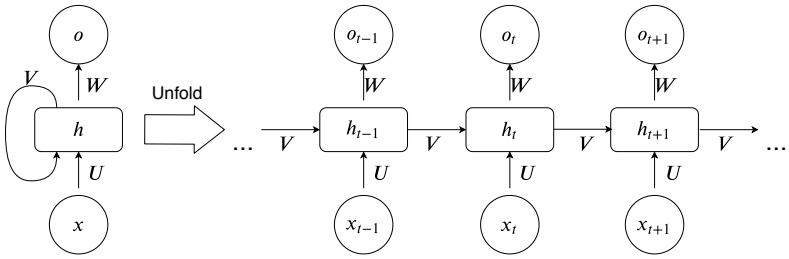


Figure 3.4: Left: Folded RNN. Right: Unfolded RNN.

Also, RNNs are flexible in terms of input-output size, this means that since they work timestep by timestep, the RNNs could be applied to different input lenghts. Further, the output could be a sequence (each of the outputs of a layer) or a vector (the last value of the sequence) and depending of the task, is possible to feed all the sequence or part of it.

Because each hidden state is a function of the prior state, the memory present in RNNs is just one step in time. And, if that is not enough, RNNs suffer from vanishing gradients, this means that when trained with long sequences, the gradients tends to be more smaller leading to long training runs.

To overcome this problems, Long Short Term Memory (LSTM) arrives as a solution. In this models are introduced a long memory (called cell c) and the known short memory (called state h) which controls the information that has to be deleted and the one who is favourable to keep, since it has three hidden transformations, the gradients are less likely to vanish.

To manage the memory in the layer, the three principal transformations are:

- **Forget gate** f_t: This extracts the long term information that is expired and deletes it.
- **Input gate** i_t: This gate choose the new information to be added to the long term from the previous hidden state and the current input data.
- **Output gate** o_t: Finally, the output gate, fuses the long memory with selected new data generating the new hidden state.

Having regard to the above, a diagram of the cell is provided:

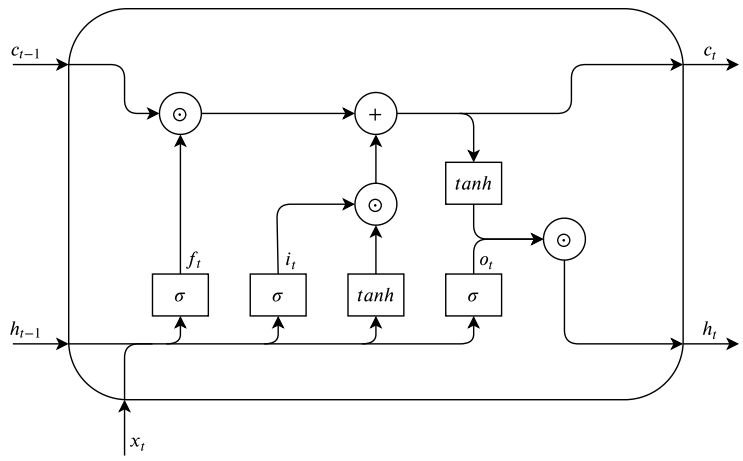


Figure 3.5: LSTM cell.

The equations below describe the operation inside the cell.

$$f_t = \sigma(U_f \cdot h_{t-1} + W_f \cdot x_t + b_f) \tag{3.8}$$

$$\mathrm{i}_t = \sigma(U_\mathrm{i} \cdot h_{t-1} + W_\mathrm{i} \cdot x_t + b_\mathrm{i}) \tag{3.9}$$

$$o_t = \sigma(U_o \cdot h_{t-1} + W_o \cdot x_t + b_o) \tag{3.10}$$

$$c_t = f_t \odot c_{t-1} + \mathrm{i}_t \odot \tanh(U_c \cdot h_{t-1} + W_c \cdot x_t + b_c) \tag{3.11}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.12}$$

However, this number of operations tends to affect the training time. This is why Gated Recurrent Units (GRU) studies which are the less important gates in favor to eliminate operations and optimize the learning process.

GRUs are constructed with:

- **Reset gate**: Drops information that is found to be irrelevant at the future.
- **Update gate**: Controls the information that pass from the previous hidden state to the current hidden state.

With this, in the 3.6 are a GRU cell:

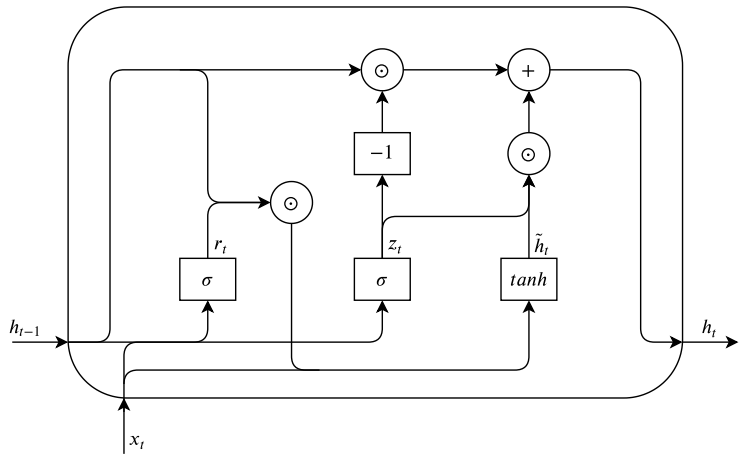


Figure 3.6: GRU cell.

$$z_t = \sigma(U_z \cdot h_{t-1} + W_z \cdot x_t + b_z) \tag{3.13}$$

$$r_t = \sigma(U_r \cdot h_{t-1} + W_r \cdot x_t + b_r) \tag{3.14}$$

$$\tilde{h}_t = \tanh(U_{\tilde{h}} \cdot (r_t \odot h_{t-1}) + W_{\tilde{h}} \cdot x_t + b_{\tilde{h}}) \tag{3.15}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \cdot \tilde{h}_t \tag{3.16}$$

As it can be seen, GRU does not have memory units, it just uses gates to modulate the flow of information [29]. The procedure is similar to LSTM cells but exposing the whole hidden state each time. Another difference is that GRUs controls the information flow at the same time that computes the new candidate whereas LSTM controls it with the output gate.

3.2.4 Training of Neural Networks

The training of the neural networks consists in optimize the parameters θ (that includes weights and biases) to generate a "good" map function $\hat{y} = f(x)$ where x is the input and $\hat{y}$ the trainable map function.

As explained, in Section 3.1 machine learning consists in a training phase and a testing phase. It is clear that in training phase, the network search in an optimized way the best parameters to generate the map function, and then, at the testing phase, the weights are frozen to make predictions.

The optimization is done by computing the gradients of each parameter θ with respect to the loss function L with the backpropagation algorithm and then use some gradient descent based optimization to update each weight.

The backpropagation algorithm simply consists in the calculation of the gradients from the last layer and propagates its values across the network to the first layer with also the importance of the contribution for each neuron, which enforces the network to organize the neurons with different focuses and complement their features.

Finally, for the regression task the common used loss function is the cross-entropy, which compares the ground true with the predicted distribution output (softmax) as show in Equation 3.1:

$$L_{cross} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c) \tag{3.17}$$

On the other hand, regression uses the RMSE as shown in Equation 3.2.

3.3 Bayesian inference and Bayes By Backprop

Now that all the frequentist machine learning approach has been summarized, it is time to explain the Bayesian concepts.

Firstly, simple definitions has to be made in the context of this thesis:

- **Probability**: Is a measure of how likely an event is to occur.
- **Random variable**: Is a named situation that could result in many different events, so, it has no determined value but a probability distribution of it possibilities.
- **Probability distribution**: Is the summary of the outputs of a random variable and their probabilities which is expressed by a function.
- **Probability density (mass) function**: A function whose integral (sum) gives the probability that the value of the variable lies within the same interval. Density is used for continuous random variable and mass for discrete ones.
- **Expected value** $E[x]$: The expected value is the integral of the random variable with respect to its probability. In simple words is the mean of a large size of samples from the random variable.
- **Sampling**: Is the action of draw a value from a probability distribution.
- **Categorical distribution**: Is a probability distribution for discrete random variables. This distribution is used to determine the probabilities of belonging to a certain class. Therefore, is used as output in classification tasks.
- **Normal distribution**: Is a probability distribution for continuous (real valued) random variables. They are many more probability distributions for continuous random variables but this is the most common and used one. The main characteristic of this distribution is the bell-shaped form of its probability density function and their symmetry.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.18}$$

- **Joint distribution**: Is a probability distribution over multiple variables.
- **Conditional probability**: Is the probability of a random variable given another random variable particular value.

$$p(X|Y=y) = \frac{p(X, Y=y)}{p(Y=y)} \tag{3.19}$$

- **Chain rule of probability**: Is the conditional distribution applied to the joint distribution.

$$p(X,Y) = p(X,Y)\frac{p(Y)}{p(Y)} = p(X|Y)p(Y) \tag{3.20}$$

- **Bayes' theorem**: All the above definitions are needed to formulate this. The conditional probability of X given data Y is the posterior probability and the conditional probability of Y given X the likelihood or model evidence of data fitting. Then, $p(X)$ is the belief, also called prior probability. Finally $p(Y)$ is the data distribution.

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)} \tag{3.21}$$

- **Likelihood**: Is a measure of how well the data summarized the parameters in the model. The negative log likelihood is commonly used as a cost function in Bayesian neural networks.
- **Inference**: Is the process of computing probability distributions over certain random variables, usually after knowing values for other variables.

Suppose that a model ($p(w|D)$ with D as data and w as weights) is successfully trained and then it is desired to make and inference (y^*) for a new datapoint (x^*). So, is necessary to look the joint distribution :

$$p(y^*|x^*, D) = \int p(y^*|x^*, w)p(w|D)\mathrm{d}w \tag{3.22}$$

It is noticeable that the integral over the weights is intractable since it is taking an expectation under the posterior distribution on weights which is equivalent to using an ensemble of infinity numbers of neural networks. Then, it is possible to define an approximate distribution $q(w|\theta)$ that has to be as similar as possible to $p(w|D)$. The computation of this approximate distribution is known as variational inference since θ are the variational parameters that makes possible the inference process.

To evaluate if those two distributions are similar, the Kullback-Leibler (KL) divergence is computed, so, the optimization problem is to find the θ that minimizes the divergence between both distributions.

$$\theta_{opt} = arg\min_{\theta} KL[q(w|\theta)||p(w|D)] \tag{3.23}$$

Then, the definition of the KL divergence is:

$$KL[q(w|\theta)||p(w|D)] = \int q(w|\theta)\log\frac{q(w|\theta)}{p(w|D)}\mathrm{d}w \tag{3.24}$$

Again, it came out another integral over the weights, so this is not solvable. To deal with the above we define the Expectation with respect to a distribution as:

$$\mathbb{E}_{q(w|\theta)}[f(w)] = \int q(w|\theta)f(w)\mathrm{d}w \tag{3.25}$$

Also, the conditional distribution for $p(w|D)$ could be replaced by the Bayes' theorem, leading to:

$$KL[q(w|\theta)||p(w|D)] = \mathbb{E}_{q(w|\theta)}\left[\log \frac{q(w|\theta)p(D)}{p(D|w)p(w)}\right] \tag{3.26}$$

Finally, splitting the logarithms and doing some algebra:

$$KL[q(w|\theta)||p(w|D)] = \mathbb{E}_{q(w|\theta)}\left[\log \frac{q(w|\theta)}{p(w)} - \log p(D|w) + \log p(D)\right] \tag{3.27}$$

It is possible to extract the $\log p(D)$ since it is constant for the integral and the first logarithm term is the KL divergence between $q_\theta(w|D)$ and $p(w)$, leading to:

$$KL[q(w|\theta)||p(w|D)] = KL[q(w|\theta)||p(w)] - \mathbb{E}_{q(w|\theta)}[\log p(D|w)] + \log p(D) \tag{3.28}$$

Back to Equation 3.23, since this is a minimization problem, is possible to ignore the $\log p(D)$ as it is a constant.

$$\begin{aligned}\theta_{opt} = arg\min_\theta KL[q(w|\theta)||p(w|D)] &\sim arg\min_\theta KL[q(w|\theta)||p(w)] - E_{q(w|\theta)}[\log p(D|w)] \\ &\sim arg\min_\theta -ELBO(D,\theta)\end{aligned} \tag{3.29}$$

Which is known as the negative of the Evidence Lower Bound (ELBO), thus, maximizing the ELBO is the same as minimizing the KL between the distribution of interest.

The ELBO contains a data dependant part which is the likelihood cost and a prior dependant part which is the complexity cost. As is a subtraction between these two terms, the loss is a trade-off between complexity of the data and the simplicity prior. However, this is finally computable by unbiased Monte Carlo gradients and backpropagation [30].

First, it is necessary to get the gradient of an expectation. It is possible to define:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(w|\theta)}[f(w,\theta)] = \mathbb{E}_{q(\varepsilon)}\left[\frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial\theta} + \frac{\partial f(w,\theta)}{\partial\theta}\right] \tag{3.30}$$

Where ε is a random variable with probability density $q(\varepsilon)$ and $w = t(\theta,\varepsilon)$ where $t(\theta,\varepsilon)$ is a deterministic function. Also, as further supposition $q(\varepsilon)\mathrm{d}\varepsilon = (w|\theta)\mathrm{d}w$.

The above deterministic function transforms a sample of parameter-free noise ε and the variational posterior parameters θ into a sample from the variational posterior. With this, it is possible to draw a sample with Monte Carlo to evaluate the expectations, enabling to do a backpropagarion like algorithm.

This backpropagation (Bayes by Backprop) approximate the exact cost as:

$$-ELBO = \sum_{\mathrm{i}=1}^{n} \log q(w^{(\mathrm{i})}|\theta) - \log p(w^{(\mathrm{i})}) - \log p(D|w^{(\mathrm{i})}) \tag{3.31}$$

Where the superindex denotes the ith Monte Carlo sample drawn from the variational posterior $q(w^{(\mathrm{i})}|\theta)$. This depends upon the particular weights drawn from the variational posterior (common random numbers) [31].

However, Bayes by backprop is a generalization for the Gaussian reparameterization trick used for latent variables models in Bayesian context (also called Variational Bayesian neural nets) which contains the problem that all weights in a batch share the same weight perturbation limiting the variance reduction effect of large batches. To deal with this, Flipout is an efficient method to avoid this problem.

3.3.1 Weight perturbations

The reparameterization trick is a weight perturbation method, which are all the methods that sample the weights of a neural network stochastically at training time.

This methods consider that the minimization of an expected loss $\mathbb{E}_{(x,y)\sim D, W\sim q_\theta}[\mathcal{L}(f(x,W),y)]$ and q_θ could be described as perturbations to the weights $W = \bar{W} + \Delta W$ where $\bar{W}$ are the mean weights and ΔW a stochastic perturbation.

In addition to addressed Variational Bayesian neural nets, most known methods are:

- **Gaussian perturbations**: Being a sample $W_{\mathrm{i}j} \sim \mathcal{N}(\bar{W}_{\mathrm{i}j}, \sigma^2_{\mathrm{i}j})$, then, using the reparameterization trick [32] this sample can be rewritten as $W_{\mathrm{i}j} = \bar{W}_{\mathrm{i}j} + \sigma_{\mathrm{i}j}\varepsilon_{\mathrm{i}j}$ where $\varepsilon \sim \mathcal{N}(0,1)$ allowing the algorithm to compute the gradients by backpropagation. Another variation is to sample from $W_{\mathrm{i}j} \sim \mathcal{N}(\bar{W}_{\mathrm{i}j}, \bar{W}^2_{\mathrm{i}j}\sigma^2_{\mathrm{i}j})$ (this means $W_{\mathrm{i}j} = \bar{W}_{\mathrm{i}j}(1+\sigma_{\mathrm{i}j}\varepsilon_{\mathrm{i}j})$) to make the information content of the weights invariable with the scale.
- **DropConnect**: This is a regularization inspired by dropout which randomly change the values of a subset of the weights as zeros.
- **Evolution strategies**: This methods works as black box optimization algorithms and nowadays are proposed as reinforcement learning algorithm. In each iteration the method generate a collection of weight perturbations and evaluates it performance.

Finally, the weight perturbation method applied in this thesis is the Flipout.

3.3.1.1 Flipout

In Section 3.3, Flipout is introduced as an efficient method to deal with the same perturbation across a batch problem, which leads into higher variances.

This works with two main assumptions about the distribution q_θ. This is that the perturbation of different weights are independent and is symmetric around zero. The above allows to do element wise multiplication without modifying the perturbation distribution.

Then, if a sample $\widehat{\Delta W} \sim q_\theta$ is multiplied by a random sign matrix (S) independent from $\widehat{\Delta W}$ then $\Delta W = \widehat{\Delta W} \odot S$ is identically distributed to $\widehat{\Delta W}$ and their gradients are also identically distributed.

With this, it is possible to draw a base perturbation $\widehat{\Delta W}$ shared in the batch and multiplies by different sign matrix, generating an unbiased estimator for the loss gradients.

$$\Delta W_n = \widehat{\Delta W} \odot v_n s_n^\top \tag{3.32}$$

Where r_n and s_n are random vectors sampled uniformly from ± 1.

Then, the activations of a dense layer are:

$$\begin{aligned} y_n &= \phi(W^\top x_n) \\ &= \phi\left((\overline{W} + \widehat{\Delta W} \odot v_n s_n^\top)^\top x_n\right) \\ &= \phi(\overline{W}^\top x_n + (\widehat{\Delta W}^\top (x_n \odot s_n)) \odot v_n) \end{aligned} \tag{3.33}$$

Which is vectorizable and defines the forward pass. Since V and S are sampled independently it is possible to obtain derivatives of the weights and the input allowing to do backpropagation.

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

$$\overline{W}_{t+1} = \overline{W}_t + \alpha \frac{1}{MN\sigma^2} \sum_{m=1}^{M} \sum_{n=1}^{N} F_{mn} \left\{ \widehat{\Delta W}_m \odot v_{mn} s_{mn}^\top \right\} \tag{3.34}$$

Where, N are the flipout perturbations at each worker, M the number of workers and F_{mn} the reward at the nth perturbation at worker m.

Chapter 4

Proposed approach of Bayesian Recurrent Neural Networks

This Chapter includes the proposed Bayesian Recurrent Neural Networks for classification, RUL and HI prognosis with a graph of the framework to construct the B-RNN and the new cell operations.

4.1 Proposed framework

The Figure 4.1 shows the construction of a B-RNN model to address RUL prediction with all their important flows from the dataset and the mathematical background.

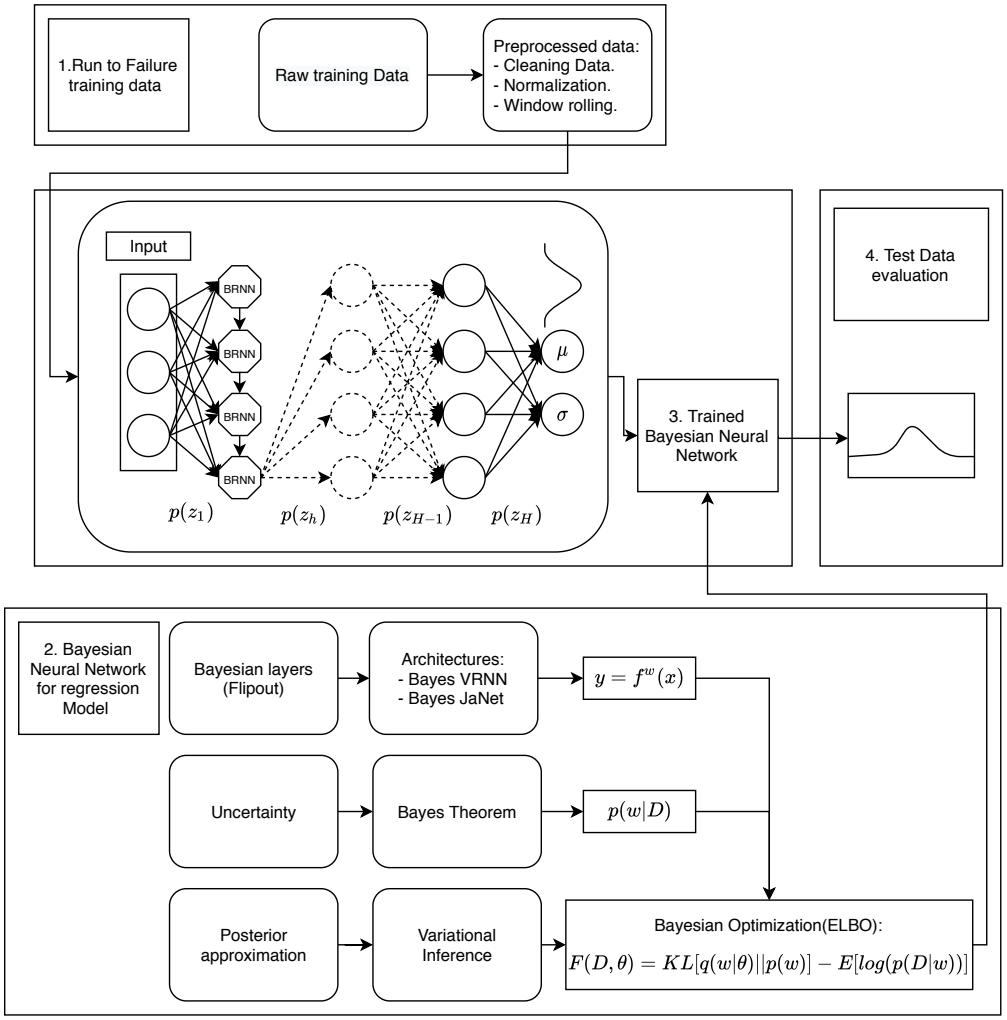


Figure 4.1: B-RNN construction flowchart.

The first step to construct a Bayesian RNN is to have proper data to work, since the approach uses recurrent neural networks, it is necessary to use sequential data. Depending of the type of dataset the task is selected i.e. if it considers run to failure data, RUL prognosis is feasible, if they are different failures at different files, classification, also it is possible to generate a health indicator supposing that none of them are provided based into the time in operation or a statistic feature that grows (or shrinks) monotonically [33].

To enhance the performance of the model, it is possible to split randomly the dataset into a test-train set but this could lead into an over-performance since the develop works with sequential data and this is essential to implement this kind of model in real environment. The above implies that the data to train are sensory data accumulated over time until present and the new incoming sensor data are the ones to test the performance.

Then, the dataset must be preprocessed with a cleaning stage where the out of scale and NaNs are dropped out from the data. Normalization, to enforce the model to learn without bias over the scale of each particular sensor and window rolling to shape correctly the dataset. In this process of windowing, it is possible to do it directly over the sensor or by features extracted from previously created windows, both methods are used in different applications at this thesis.

Later, the model is created by the utilization of LSTM or GRU cells as Keras layers, which enables an easy way to create the model. However, since this are custom made layers which contains dense flipout layers, a custom made training function is applied. With this, and the optimization of the ELBO as loss function, an output distribution is trained, from where it is possible to draw any number of samples for a chosen input which is the final prediction.

4.2 Cell operations

The proposed approach of Bayesian Recurrent Neural Networks consists in the transformation of the inner gates calculation of the frequentist RNN cells into dense layers and then, convert them into Bayesian layers by the application of the Flipout scheme.

The first step, is transform the operations inside the RNN cells to dense layers, for example, the update gate of the GRU layer:

$$z_t = \sigma(U_z \cdot h_{t-1} + W_z \cdot x_t + b_z) \tag{4.1}$$

Where multiplications between the weights and the inputs could be replaced by dense layers for each input (hidden state and new data):

$$z_t = \sigma(Dense(h_{t-1}) + Dense(x_t)) \tag{4.2}$$

Then, each dense layer is transformed to a Bayesian layer by the application of Flipout, resulting:

$$z_t = \sigma\left(\left[\overline{U_z}h_{t-1} + (\widehat{\Delta U_z}(h_{t-1} \odot s^h_{z,t})) \odot v^h_{z,t}\right] + \left[\overline{W_z}x_t + (\widehat{\Delta W_z}(x_t \odot s^x_{z,t})) \odot v^x_{z,t}\right]\right) \tag{4.3}$$

Where a sub-index z, t represents that is for gate z in time t and a super-index that represent if it for a hidden state h or an input x. Also, since an activation function for each dense layer is not necessary, it just left with an identity activation function.

It is important to notice than this notation includes the sample of the sign vectors at each time step.

With the above, it is possible to implement the proposed approach for all the remaining gates as follows:

Bayes LSTM Operations
Using the same steps as above, LSTM gates are defined by:

$$f_t = \sigma(\overline{U_f} h_{t-1} + (\widehat{\Delta U_f}(h_{t-1} \odot s^h_{f,t})) \odot v^h_{f,t} + \overline{W_f} x_t + (\widehat{\Delta W_f}(x_t \odot s^x_{f,t})) \odot v^x_{f,t}) \tag{4.4}$$

$$\mathrm{i}_t = \sigma(\overline{U_\mathrm{i}} h_{t-1} + (\widehat{\Delta U_\mathrm{i}}(h_{t-1} \odot s^h_{\mathrm{i},t})) \odot v^h_{\mathrm{i},t} + \overline{W_\mathrm{i}} x_t + (\widehat{\Delta W_\mathrm{i}}(x_t \odot s^x_{\mathrm{i},t})) \odot v^x_{\mathrm{i},t}) \tag{4.5}$$

$$o_t = \sigma(\overline{U_o} h_{t-1} + (\widehat{\Delta U_o}(h_{t-1} \odot s^h_{o,t})) \odot v^h_{o,t} + \overline{W_o} x_t + (\widehat{\Delta W_o}(x_t \odot s^x_{o,t})) \odot v^x_{o,t}) \tag{4.6}$$

$$c_t = f_t \odot c_{t-1} + \mathrm{i}_t \odot \tanh(\overline{U_c} h_{t-1} + (\widehat{\Delta U_c}(h_{t-1} \odot s^h_{c,t})) \odot v^h_{c,t} + \overline{W_c} x_t + (\widehat{\Delta W_c}(x_t \odot s^x_{c,t})) \odot v^x_{c,t}) \tag{4.7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{4.8}$$

Bayes GRU Operations
Using the same steps as above, GRU gates are defined by:

$$z_t = \sigma(\overline{U_z} h_{t-1} + (\widehat{\Delta U_z}(h_{t-1} \odot s^h_{z,t})) \odot v^h_{z,t} + \overline{W_z} x_t + (\widehat{\Delta W_z}(x_t \odot s^x_{z,t})) \odot v^x_{z,t}) \tag{4.9}$$

$$r_t = \sigma(\overline{U_r} h_{t-1} + (\widehat{\Delta U_r}(h_{t-1} \odot s^h_{r,t})) \odot v^h_{r,t} + \overline{W_r} x_t + (\widehat{\Delta W_r}(x_t \odot s^x_{r,t})) \odot v^x_{r,t}) \tag{4.10}$$

$$\tilde{h}_t = \tanh(\overline{U_{\tilde{h}}}(h_{t-1} \odot r_t) + (\widehat{\Delta U_{\tilde{h}}}((h_{t-1} \odot r_t) \odot s^h_{\tilde{h},t})) \odot v^h_{\tilde{h},t} + \overline{W_{\tilde{h}}} x_t + (\widehat{\Delta W_{\tilde{h}}}(x_t \odot s^x_{\tilde{h},t})) \odot v^x_{\tilde{h},t}) \tag{4.11}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \tilde{h}_t \tag{4.12}$$

At this equations, the notation is:

- $\overline{W}$, $\overline{U}$: Are the mean weights matrix sampled from the distribution q_θ.
- $\hat{W}$, $\hat{U}$: The stochastic perturbation of the mean weights sampled from the distribution q_θ.
- v, s: Column vectors from V and S which are sampled uniformly from ± 1.

4.3 Training and implementation of Bayesian Neural Networks

The training of Bayesian Neural Networks consists in the application of Bayes by Backprop as backpropagation of the gradients to minimize the ELBO loss.

This loss is divided in two main parts:

- **Kullback-Leibler**: Is a regularization term for the overall loss. Also, it is many magnitude orders higher than the Log likelihood. Because of this, is necessary to multiply this number by a regularizer in search of the best tradeoff between both losses.
- **Log likelihood**: Empirically, this loss is the one who rules the performance of the network. So, the balance between KL and Log likelihood is doing well if the log likelihood is stable and the KL is not growing.

With that, the implementation consists in the application of Tensorflow Probability Flipout Dense Layers in new layers called Bayes-LSTM and Bayes-GRU wrapping their inner cell calculations. With this, the Bayes by Backprop implementation is straightforward and the output is a chosen distribution depending on the application (categorical if it is a classification and normal distribution at the regression).

4.3.1 Regression metrics

A the regression task the results are N samples from the final distribution. With this is possible to get two different RMSE, the first one is the called Mean RMSE or (RMSE) which is the mean of the RMSE of the N samples and the other is the Global RMSE (RMSE*) which is the RMSE of the mean for each point at the N samples. Since the output is the distribution and not the single samples, the RMSE* is the most comparable metric with the RMSE of the frequentist approaches with the difference that is possible to compute the deviation of the RMSE* too.

4.3.2 Classification metrics

As same as the regression task, the output of the model is a sampleable distribution, so the result are N samples from it. This samples are the probability of being from a certain class and because of this, their length is the same as the number of classes at the dataset. As it is expected, we could obtain many different confusion matrix from the above samples (due to the incorporation of the uncertainty) and then calculate a probability interval for each cell of the matrices resulting in a confusion matrix that summarizes the performance of the classifier.

Chapter 5

Study cases

In this Chapter the datasets in which this thesis is validated are described, including their setup, origin and more relevant characteristics. Also, this datasets could need some preprocess prior to be feeded in the network, so this step is also explained. Finally, after a grid search to look for the best hyperparameters the architecture of the network is fixed and is presented.

5.1 Datasets for regression tasks

The regression tasks includes the assessment of the remaining useful life (C-MAPSS) and health indicators (Maryland cracklines and wind turbine data).

5.1.1 NASA: C-MAPSS

5.1.1.1 Data description

C-MAPSS stands for Commercial Modular Aero-Propulsion System Simulator [1]. Is a software created by NASA to generate a thermo-dynamical simulation model for the turbofan engine as function of variations of flow an efficiency. This data is the most used at RUL prognosis and also is used in the PHM08 challenge. From then, they are many different approaches to obtain a prediction of the RUL and the one who is taken the most successful results is Deep Learning.

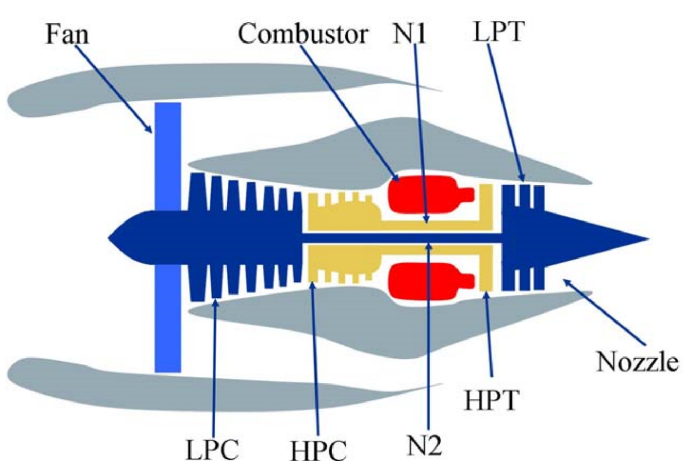


Figure 5.1: Simplified diagram of engine simulated in C-MAPSS. Source: [1].

The dataset consists in 4 different sub-datasets which are divided by the number of operation conditions and fault modes:

Table 5.1: Definition of the 4 sub-datasets by it operation conditions and fault modes.

		Operation conditions	
		1	6
Fault modes	HPC Degradation	FD001	FD002
	HPC and Fan Degradation	FD003	FD004

Where HPC stands for high-pressure compressor. In each of this datasets are 26 columns, which are:

- Unit number
- Time [cycles]
- 3 operational settings
- 21 sensor measurements

The outputs of the measure system response are:

Table 5.2: C-MAPSS output sensor data.

Sensor tag	Description	Units
1	Total temperature at fan inlet	^{o}R
2	Total temperature at LPC outlet	^{o}R
3	Total temperature at HPC outlet	^{o}R
4	Total temperature at LPC outlet	^{o}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	dimentionless
11	Static pressure at HPC outlet	psia
12	Ratio of fuel flow to Ps30	pps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	dimentionless
16	Burner fuel-air ratio	dimentionless
17	Bleed enthalpy	dimentionless
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	HPT coolant bleed	lbm/s
21	LPT coolant bleed	lbm/s

Where LPC stands for low pressure compressor, HPT for high pressure turbine and LPT for low pressure turbine.

5.1.1.2 Pre processing

First, the dataset is constructed as follows:

- Each row corresponds to one cycle.
- Each column represent a sensor or an operational setting.
- A single unit is simulated (Unit number columns) and it is under operation N cycles until failure. Then, another unit is put into operation and register its sensor data.
- They are 7 sensors that does not change with the cycles, this means that they do not deliver useful information, so, they are ignored.

Since this dataset is used for a challenge, it is already divided in train and test samples and also, the RUL value is in another file for train and test.

Therefore, it is not necessary to split the dataset and it is possible to create windows with 1 cycle as slide step without contaminating the train with test data.

Due to data conformation, not all datasets have the same window size. With this, the created datasets have the following shape:

Table 5.3: Dataset size for C-MAPSS

Dataset	Window length	Sensors	Train/Test	Number of windows
FD001	30	14	Train	14347
			Test	100
FD002	21	14	Train	29526
			Test	259
FD003	30	14	Train	17864
			Test	100
FD004	18	14	Train	46326
			Test	248

5.1.1.3 Architecture

A grid search is used to fix parameters of the model, some of the selected numbers to be tried are summarized in the following table:

Table 5.4: Table of Hyperparameters to test.

Hyperparameter	Grid search	Hyperparameter	Grid search
BayesRNN Neurons	[16, 32, 64]	Scale	[1e0, 5e0, 1e1]
DenseFlipout Layers	[1, 2, 3]	Softplus	[1e-3, 5e-3, 1e-2]
DenseFlipout Neurons	[64, 32, 16]	LR decay	[40, 60, 100]
Epochs	[50, 100, 300]	End LR	[50%, 70%]
Learning Rate	[1e-2, 1e-3, 1e-4]	KL Regularizer	[1e-5, 5e-6, 1e-6]

The final architecture to address the C-MAPSS dataset is:

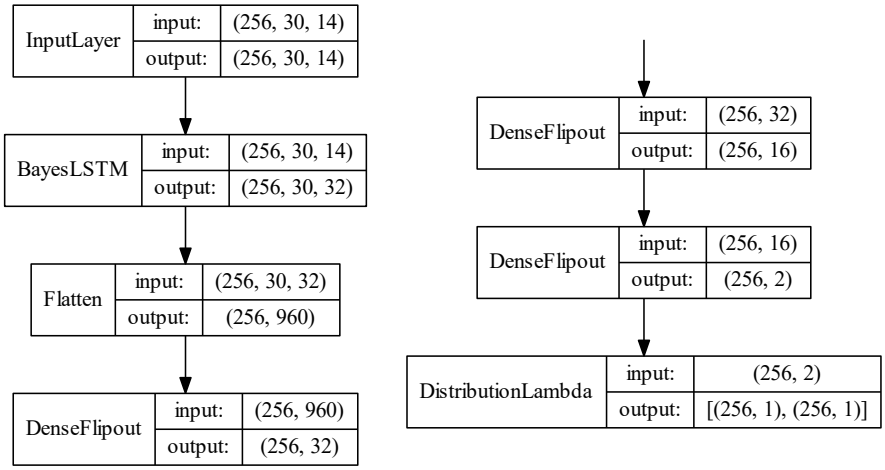


Figure 5.2: Bayesian recurrent layers architecture for the C-MAPSS dataset. In particular, with BayesLSTM layer.

Notice that the output of the Bayesian RNN is the complete sequence and the flatten layer prepares the features to be processed by the dense flipout layers.

Also, a decaying polynomial learning rate is applied to smooth the learning in the last epochs, the learning rate decays in 60 epochs to the 70% of the learning rate at the epoch 1.

A summary of the hyperparameters is presented in the Table 5.5:

Table 5.5: Hyperparameters for the C-MAPSS dataset.

Hyperparameter	Value	Hyperparameter	Value
Batch size	256	KL regularizer	1e-6
Epochs	100	KL annealing [epochs]	20
Scale	1e1	Learning rate	0.001
Softplus	1e-3	End learning rate	70%
Learning rate decay [epochs]	60		

5.1.2 Maryland: Crack Lines

5.1.2.1 Data description

This dataset is from a Doctoral thesis developed at the University of Maryland [34] and consists in the acquisition of three dissipative energies (strain, heat dissipation and acoustic emission) during a cycle tensile uniaxial test.

With the above energies is possible to quantify the damage in specific failure modes by the Degradation-entropy generation (DEG) theorem which is introduced in [35, 36, 37] and reviewed in applications including fatigue, corrosion and wear [38]. Also, the crack length is monitored with images taken by an Edmond 2.5-10x optical microscope with an OptixCam Pinnacle Series CCD digital camera. This with the corresponding life in cycles define the life of failure at a certain crack length. 250[μm], 500[μm], 1000[μm], the transition from region II to III (with linear elastic fracture mechanics [39]) and fracture are the selected failure determinations for each entropic approach's assessment.

Two different materials are selected as testing materials, AA7075-T6 and SS304L. The first one is an aluminum alloy with high-strength and light, the other is an structural material used commonly in highly acidic environments. Since the conformation of the dataset, the only measurements that have enough datapoints to work with the B-RNN scheme is the stainless steel

Table 5.6: Mechanical properties of SS304L.

Mechanical properties			
σ_{UTS}[MPa]	σ_y[MPa]	Elongation [%]	Hardness [RB]
613.8	325.7	54.06	85.00
Chemical composition [w%]			
C	Cr	Cu	Mn
0.0243	18.06	0.3655	1.772
Mo	N	Ni	P
0.2940	0.0713	8.081	0.0300
S	Si		
0.0010	0.1930		

The specimen is selected and designed for fatigue testing under ASTM 466 guideline [40].

An Instron 8800 system on an MTS 311.11 frame is used to mount the uniaxial test. The loads are in the range of 16 to 24 [kN], 0.1 loading ratio and 5 [Hz] frequency. For applications to short load term process every 1000 cycles, the load is paused by 2 minutes and 500 cycles of excitation are applied at 6 [kN] for the aluminum and 10 [kN] for the stainless steel as maximum loads at 25 [Hz]. The time without load allowed the specimen's cooling and it is the perfect time to capture cleaner images. The acoustic emission sensor are two Physical Acoustics Micro-30s resonant sensors.

5.1.2.2 Pre processing

With this, the dataset consist in 3 signal measurements (position, extension and temperature) and 4 cumulative measurements (2 acoustic emission counts and 2 acoustic emission energy). The proposed features to extract consider the Crest factor and the mean of the Fourier transform splitted in frequency bands, this two features are not feasible with the last 4 measurements due to its behavior, even so, peak to peak value has no useful information.

Due to the above, there are many ways to work this dataset, the tested approaches are listed below:

Table 5.7: Tested datasets for Maryland crack lines.

Raw/Features	Columns	Feature excluded	Total features per column	Time dimension	Feature dimension
Features	Every measurement	Crest factor, all fft	24	20	168
Features	Only sensor data	fft of derivative and integral	32	20	96
Raw	Only sensor data	-	-	200	3
Features	Only cummulative data	Crest factor, all fft	24	20	96
Raw	Only cummulative data	-	-	200	4

It is important to notice that the raw windows have length of 200 due to the composition of the extracted features which have a 10 length window to perform the extraction and then 20 length window to train the RNN, creating a dataset with the same numbers of examples.

After testing all this datasets, the best results are obtained with only the sensor data and the samples are constructed as follows:

Table 5.8: Dataset size for Maryland cracklines.

Feature window length	Features	Window length	Train/Test	Number of windows
10	96	20	Train	3844
			Test	646

The test set consists in 7 of 47 total runs, which is almost a 15% split.

5.1.2.3 Architecture

The grid search considers the following hyperparameters:

Table 5.9: Hyperparameters tested in the grid search

Hyperparameter	Grid search	Hyperparameter	Grid search
Conv Layers	[1, 2, 3]	Conv N of kernels	[16, 32, 64]
Conv kernels (only time)	[1, 2, 3]	LR decay	[10, 20, 30]
BayesRNN Neurons	[16, 32, 64]	End LR	[10%, 50%]
DenseFlipout Layers	[1, 2, 3, 4]	KL Regularizer	[1e-5, 5e-6, 1e-6]
DenseFlipout Neurons	[64, 32, 16]	Epochs	[1000, 2000, 3000]
Learning Rate	[1e-2, 1e-3, 5e-4]		

The final architecture selected is an ensemble of two frecuentist 2-dimensional convolutional layers, with kernels that only works at the sensor dimension, to work as feature extractor of the raw sensor data. Then, a time distributed layer reshapes the data to be ready for the Bayesian RNN layers, then flatten and dense flipout to deal with the regression.

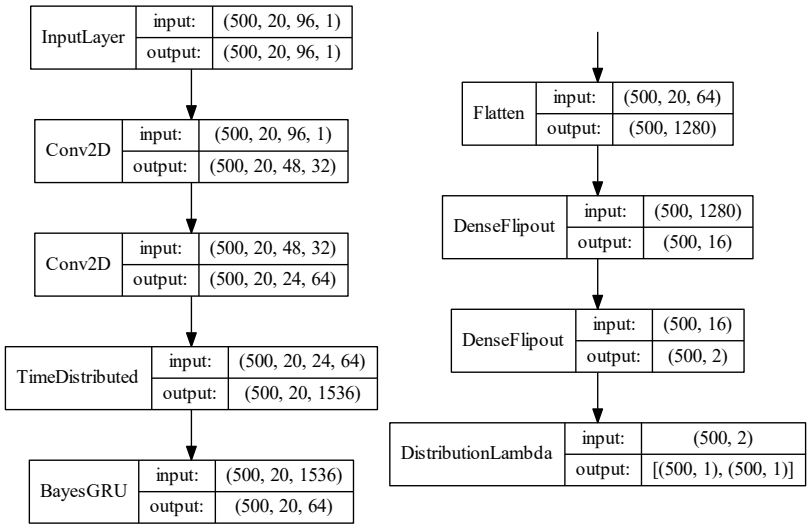


Figure 5.3: Bayesian recurrent layers architecture for the Maryland cracklines dataset.

Table 5.10: Hyperparameteres for the Maryland dataset.

Hyperparameter	Value	Hyperparameter	Value
Batch size	500	KL Regularizer	1e-5
Epochs	3000	KL annealing [epochs]	20
Learning rate	0.0005	End learning rate	10%
Learning rate decay [epochs]	20		

5.1.3 GPM Systems: Wind Turbine Data

5.1.3.1 Data description

This data was collected from a commercial 2.2 [MW] wind turbine over 50 days, 6 seconds each with a sampling frequency of 100 [kHz] for a high speed shaft bearing from a wind turbine generator provided by Green Power Monitoring Systems in the USA [2, 41].

The bearing is a SKF 32222 J2 tapered roller bearings with an inner race fault (detected with inspection at the end of the 50 days). The recorded sensor is a MEMS accelerometer (analog devices ADXL001 with 32 [kHz] bandwith).

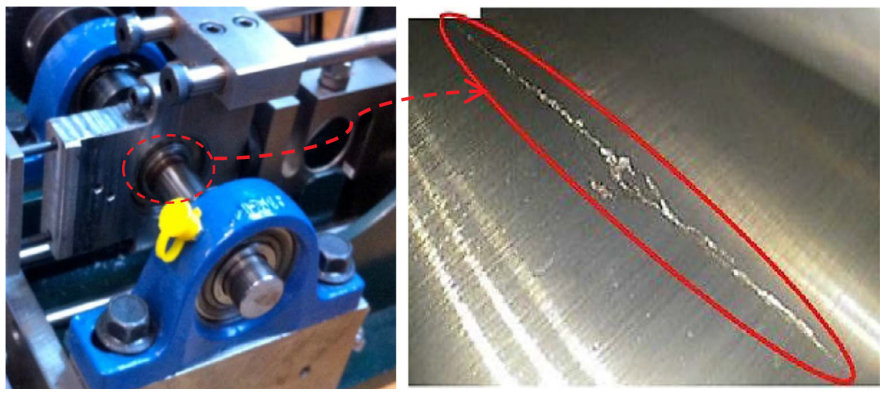


Figure 5.4: Bearing test set with the inner race fault detected at the end of the 50 days. Source: [2]

It is possible to see the degradation of the wind turbine with the growing size of the acceleration. The following Figure 5.5 visualizes all the 50 days one next to the other, even if they are many hours between each measurement:

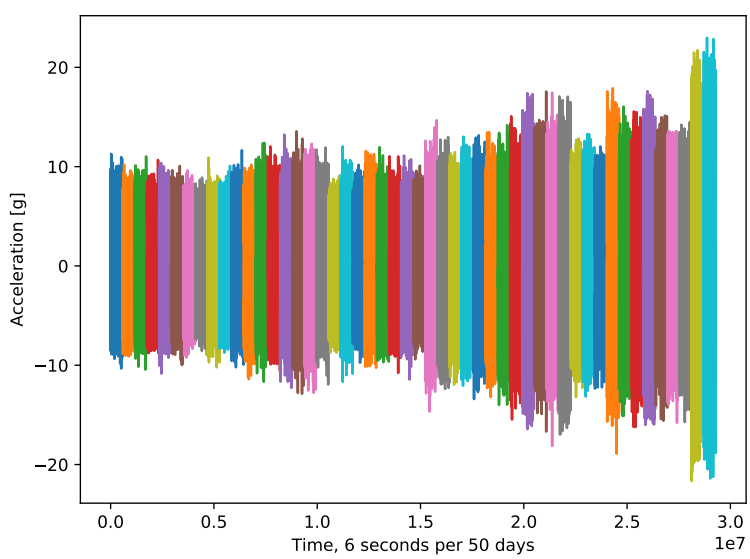


Figure 5.5: Bearing test set with the inner race fault detected at the end of the 50 days. The colors shows different days of measurements.

5.1.3.2 Pre processing

Since the dataset only consider one sensor, our approach is to extract features from the accelerometer signal, the features are presented in the Appendix A

The features are computed for the signal, numeric derivative and numeric integration, generating 42 features. To compute the features it is needed to generate windows. In our approach 200 timesteps generate 1 feature example and 50 feature example a window for training the network.

Also, this dataset is not previously splitted into train and test. Therefore, is not possible to generate the windows with overlap due to train/test contamination.

With this, for each day of 585936 datapoints 58 training windows are generated with length of 50 and 200 as feature length, the remaining of each file is dropped. Finally, a 30% test - 70% train random split is used to divide the samples.

The Figure 5.6 illustrates this dimensional reduction by the application of the preprocessing:

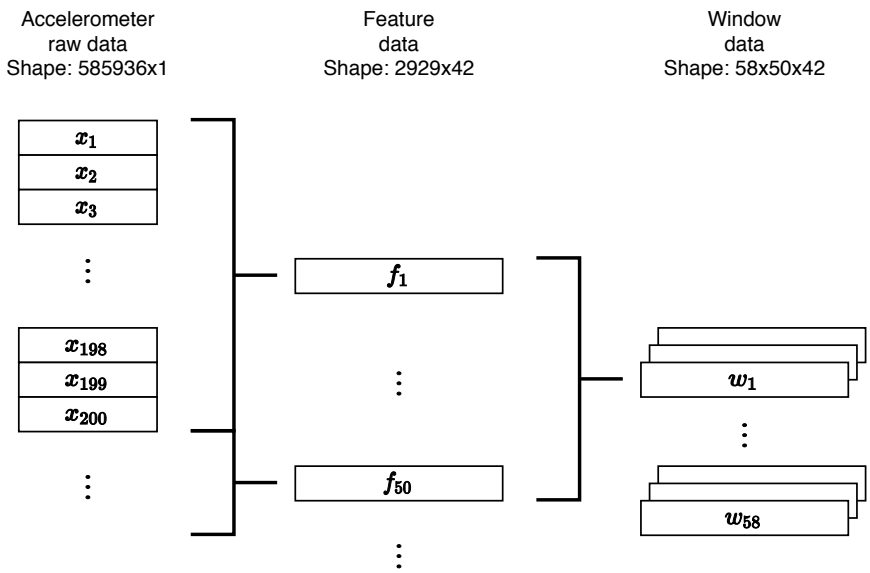


Figure 5.6: Graphical view of the dimensional reduction by preprocessing.

A summary of the dataset is presented in the Table 5.11:

Table 5.11: GPM turbine dataset shape post processing.

Feature window length	Features	Window length	Train/Test	Number of windows
200	42	50	Train	2030
			Test	870

On the other hand, the labels are not provided and the sensor data is not from the beginning of the lifespan until the failure. Therefore, it is not possible to consider the timeline as a RUL and it is needed to create a health indicator (HI). The most used ones is to find some statistic feature that changes monotonically with time and select it as HI. Since our approach is to lower the human based decisions, our HI is a decreasing line from 100 to 0, where 100 means a fully operative wind turbine and 0 the last day of operation, being this the direct approach to a HI and could be comparable to a RUL.

5.1.3.3 Architecture

First, a hyperparameter selection is made through a grid search, the most important tested parameters are listed below:

Table 5.12: Grid search for the hyperparameter selection.

Hyperparameter	Grid search	Hyperparameter	Grid search
Conv Layers	[1, 2, 3]	Conv N of kernels	[16, 32, 64]
Conv kernels (only time)	[1, 2, 3]	KL Regularizer	[1e-5, 1e-6, 1e-7]
BayesRNN Neurons	[16, 32, 64]	Epochs	[1000, 2000, 3000]
DenseFlipout Layers	[1, 2, 3, 4]	Learning Rate	[1e-2, 1e-3, 5e-4]
DenseFlipout Neurons	[64, 32, 16]		

After a grid search for hyperparameter selection, the selected architecture is as follows in Figure 5.7:

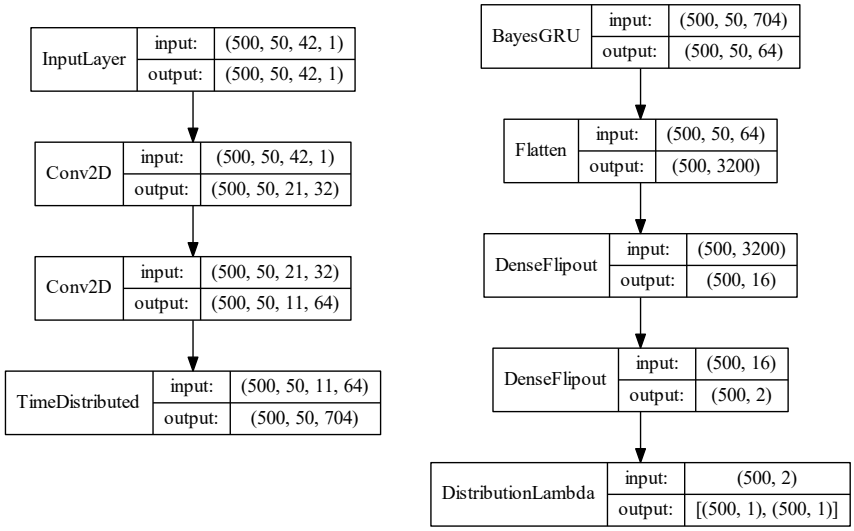


Figure 5.7: Bayesian recurrent layers architecture for the GPM Wind turbine dataset. In particular, with BayesGRU layer.

It is possible to see that the 2D convolutions have kernel and stride of size 2 in the feature index and size 1 in the time index since the Bayesian recurrent networks are the ones that deal with the temporary of the data. The activation function between every layer is an ReLU.

Also, the convolutional layers are only used as feature extractors, this allows to use frequentist approach in this layers and mix them with Bayesian weights. Following this scheme, the time distributed layer stacks the extracted features from the convolutions but maintaining the time axis.

Another important hyperparameters are:

Table 5.13: Hyperparameters for the GPM Wind turbine dataset.

Hyperparameter	Value	Hyperparameter	Value
Batch size	500	KL regularizer	1e-7
Epochs	3000	KL annealing [epochs]	20
Scale	1	Learning rate	0.001
Softplus	1e-5		

5.2 Datasets for classification tasks

The classification tasks includes the identification of different fault modes in ball bearings under time-varying speed conditions (Ottawa bearing dataset) and the health state classification for a rolling bearing under different speeds and torques (Politecnico di Torino bearing dataset).

5.2.1 Ottawa: Bearing vibration Data

5.2.1.1 Data description

This is a laboratory experimental dataset collected from bearings with different health condition under different time-varying speed conditions.

The bearing that is monitored is an ER16K ball bearing with pitch diameter of 38.52[mm], ball diameter of 7.94[mm] and 9 balls. Which is sampled at 200[kHz] for 10[s] by an ICP accelerometer, model 623C01 and an incremental encoder (EPC, model 775) is installed to measure the rotational speed.

They are 5 different health conditions and 4 operational settings, which are:

Table 5.14: Conformation of the Ottawa bearing dataset.

Health conditions	Operational settings
Healthy	Increasing speed
Inner race fault	Decreasing speed
Outer race fault	Increasing then decreasing speed
Ball defects	Decreasing then increasing speed
Mixed faults	

The experimental setup is in Figure 5.8:

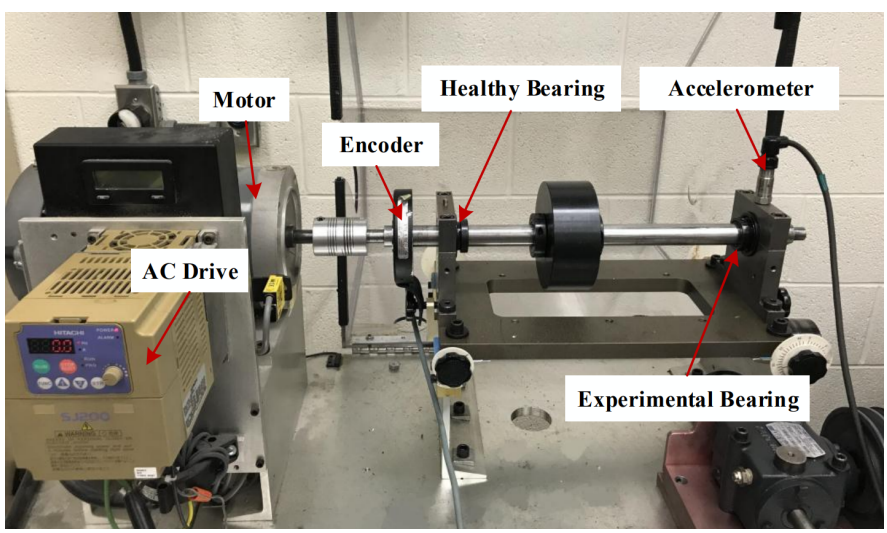


Figure 5.8: Experimental setup for Ottawa bearing dataset.

Two columns are present in the dataset, the accelerometer and the rotational speed data measured by the encoder. Also, they are 3 different runs per experiment. Therefore, they are 60 files to preprocess.

5.2.1.2 Pre processing

To preprocess this dataset, our approach is to use features extracted from the data as was done with the GPM dataset. In particular, this dataset have two columns but only the accelerometer is used.

Also, the classes used are the 5 different health conditions with all the different rotational speeds e.g. the class *inner race defect* includes examples for all rotational speeds.

As each experiment (one class, one condition) is repeated 3 times, it is possible to extract one file by experiment and train with the other two. With that, we create 3 datasets with this approach each consisting in a test set of 4 files per class and therefore, a train set with the remaining 8 files, generating a 67% train and 33% test set.

Following the same kind of preprocessing as in GPM dataset, it is needed to select a feature window size and a RNN window size. Testing with different configurations, the best behavior is obtained with 50 timesteps to generate features, and 50 features to generate a RNN example.

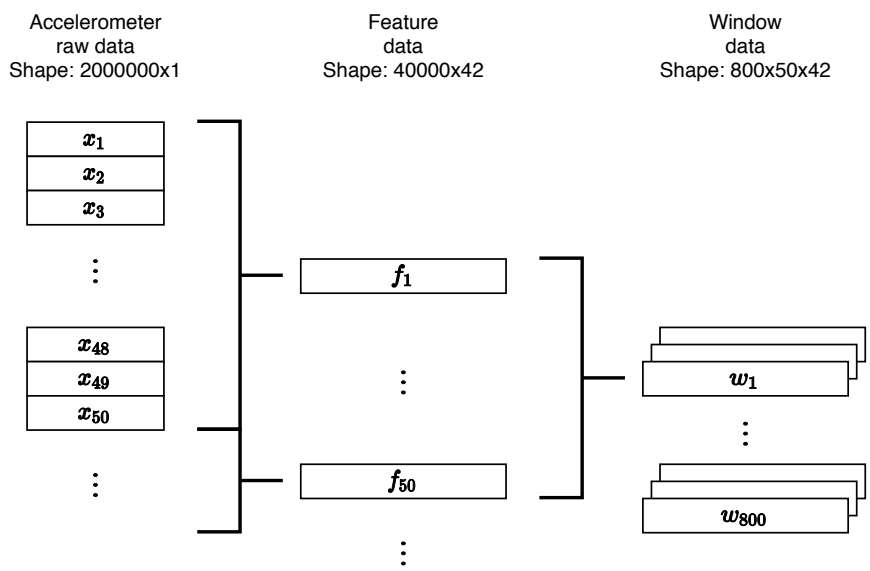


Figure 5.9: Graphical view of the dimensional reduction by preprocessing per file.

A summary of the dataset is presented in the Table 5.15:

Table 5.15: Ottawa bearing dataset shape post processing.

Feature window length	Features	Window length	Train/Test	Number of windows
50	42	50	Train	32000
			Test	16000

5.2.1.3 Architecture

A grid search based on the following table is implemented to select the hyperparameters:

Table 5.16: Hyperparameters for grid search.

Hyperparameter	Grid search	Hyperparameter	Grid search
BayesRNN Neurons	[16, 32, 64]	KL Regularizer	[1e-7, 1e-6, 1e-5]
DenseFlipout Layers	[1, 2]	Epochs	[500, 800, 1000]
DenseFlipout Neurons	[64, 32, 16]	Learning Rate	[1e-2, 1e-3, 5e-4]

After a hyperparameter selection the final architecture to address this classification task is:

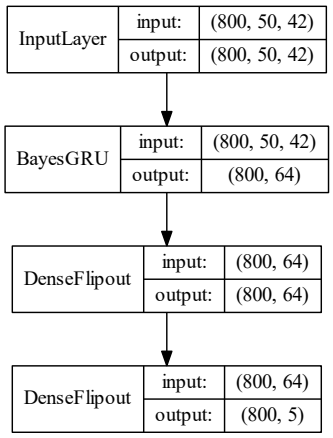


Figure 5.10: Bayesian recurrent layers architecture for the Ottawa bearing dataset. In particular, with BayesGRU layer.

Notice that the regression models outputs directly a distribution, but the classification outputs a vector (a.k.a. logits) with the number of classes as length (as in the frequentist approach). However, the softmax function is not used since the output of the model is the parameterization of a (Tensorflow probability) categorical distribution which shows the real probability of the classes.

Table 5.17: Hyperparameters for Ottawa bearing dataset.

Hyperparameter	Value	Hyperparameter	Value
Batch size	800	KL Regularizer	8e-7
Epochs	800	KL annealing [epochs]	50
Learning rate	0.001		

It is important to notice that the KL Regularizer is not at the grid search table because is fine tuned after most of the hyperparameters.

5.2.2 Politecnico di Torino: Rolling bearing Data

5.2.2.1 Data description

As Ottawa bearing dataset, Politecnico di Rotino bearing dataset is a laboratory experimental dataset of different fault modes but in rolling bearings with an orthogonal load that produces torques to the shaft.

As it can be seen in Figure 5.11, the same plate have a center bearing (B2) and two more bearings (B1 and B3) that are used as supports.

The shaft was part of a complete gearbox and carried a spur gear. This gear applies a contact force with radial an tangential directions because the applied torque. This interaction between the spur and the shaft is replaced by the load applied by a third larger roller bearing (B2) which is linked to a precision sledge that moves orthogonal to the shaft that produces the required load by the compression of two parallel springs.

The Figure 5.11 shows the experimental setup:

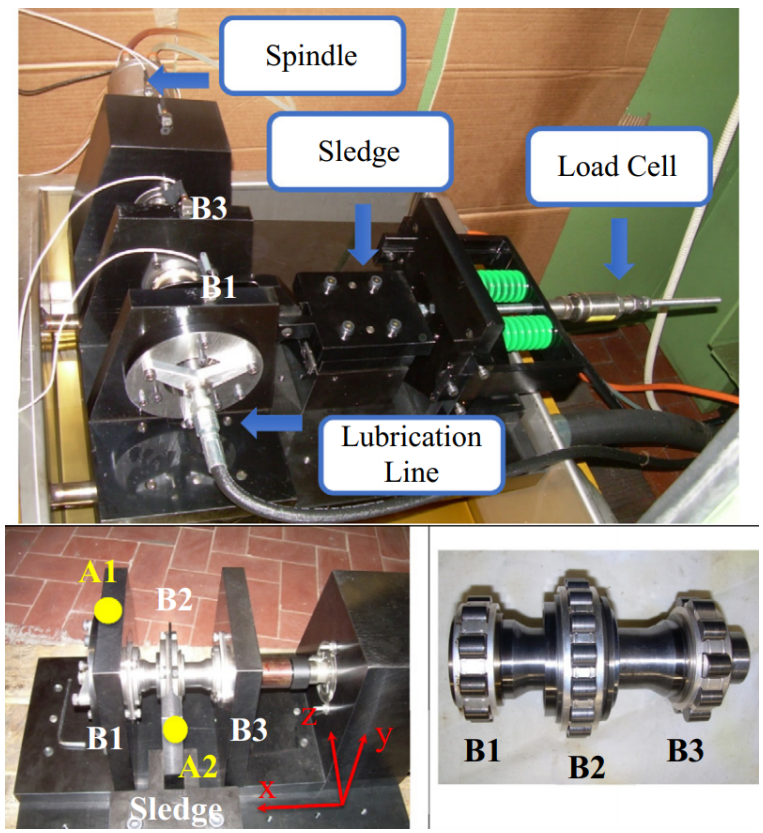


Figure 5.11: Experimental setup for Politecnico di Torino bearing dataset.

The accelerometers are two triaxial IEPE type with 12[kHz] frequency range positioned on the support of the damaged bearing (A1) and at the support of the loaded bearing (A2). Also, the radial force is measured a static load cell. The loaded bearing is the B2 but the defects are mounted in bearing B1.

The different health states are:

Table 5.18: Different health states at the B1 bearing.

Name	Defect	Diameter dimension (μm)
0A	No defect	-
1A	Inner ring	450
2A	Inner ring	250
3A	Inner ring	150
4A	Roller	450
5A	Roller	250
6A	Roller	150

The test consider the following steps:

- Brief run at minimum speed (100[Hz]) without load to check the correct mounting.
- Application of the static load: 1000[N], 1400[N] and finally 1800[N].
- Increment of the speed from 0[Hz] to 500[Hz] with steps of 100[Hz]

As soon as the shaft reach steady speed, the measurement of the accelerations are registered. The acquisition has sampling frequency of 51.2[kHz] for a duration of 10[s].

The possible configurations of speed and loads are summarized in Table 5.19:

Table 5.19: Summary of nominal loads and speeds of the shaft.

Nominal load [N]	Nominal speed [Hz]				
0	100	200	300	400	500
1000	100	200	300	400	500
1400	100	200	300	400	500
1800	100	200	300		

Due to the limited power of the inverter, at higher load conditions is impossible to reach higher rotation speeds.

5.2.2.2 Pre processing

This dataset has different speeds and torques at the shaft under 7 different failure modes. Since the goal is to classify the fault, all the files from the same health state are joined as the same class.

However, unlike Ottawa bearing dataset, Torino has only one file per torque and speed, therefore, it is not possible to split the dataset into train/test by entirely independent files. Consequently, to maintain the sequential behavior of the data, each file is divided in 70% train (the first values) and the remaining 30% as test.

Then, as they are two tri-axial accelerometers, the parameters (Appendix A) of all measurement are calculated, this is 42 features per column and a total of 252 features.

As seen before, to shape correctly the dataset it is necessary two window sizes, the first to compute the features and the last to feed the model.

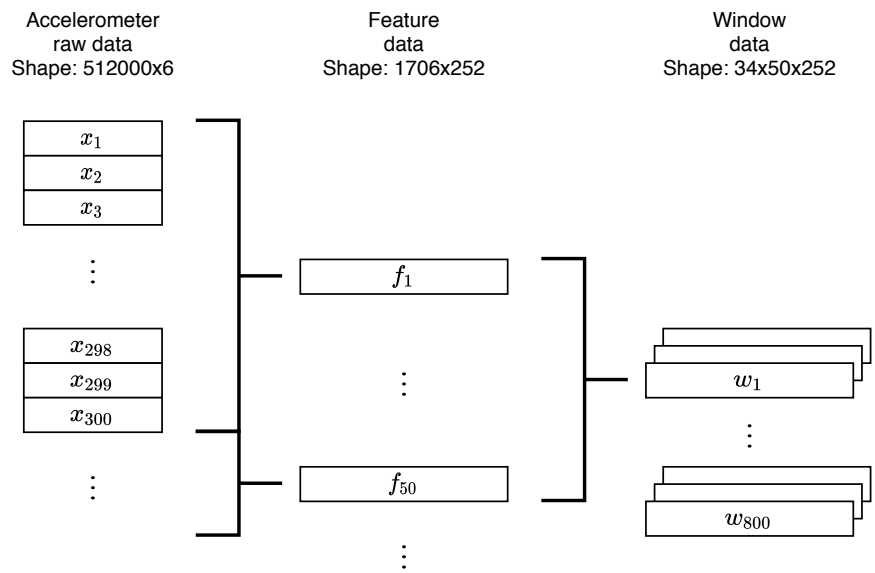


Figure 5.12: Graphical view of the dimensional reduction by preprocessing per file.

A summary of the dataset is presented in the Table 5.20:

Table 5.20: Torino bearing dataset shape post processing.

Feature window length	Features	Window length	Train/Test	Number of windows
300	252	50	Train	2856
			Test	1190

5.2.2.3 Architecture

As in the other models, the hyperparameters are selected by grid search (and then, manually fine tuned).

Table 5.21: Grid search hyperparameters.

Hyperparameter	Grid search	Hyperparameter	Grid search
BayesRNN Neurons	[16, 32, 64]	KL Regularizer	[1e-7, 1e-6, 1e-5]
DenseFlipout Layers	[1, 2]	Epochs	[300, 500, 800]
DenseFlipout Neurons	[64, 32, 16]	Learning Rate	[1e-3, 5e-3, 8e-3]
End LR	[10%, 50%]	LR decay	[40, 100]

Then, the final architecture is as follows:

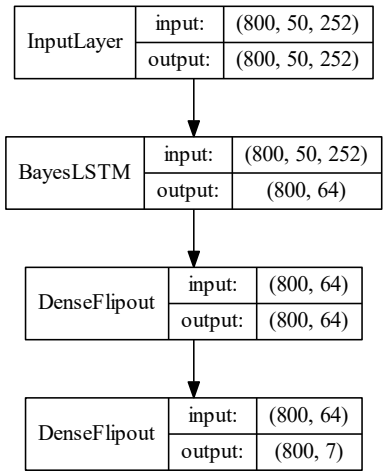


Figure 5.13: Bayesian recurrent layers architecture for the Politecnico di Torino bearing dataset. In particular, with BayesLSTM layer.

To smooth the loss curve and prevent overfit, a polynomical learning rate decay is implemented, the parameters.

The most important hyperparameters are listed in the following Table 5.22:

Table 5.22: Hyperparameters for Ottawa bearing dataset.

Hyperparameter	Value	Hyperparameter	Value
Batch size	800	KL Regularizer	5e-7
Epochs	350	KL annealing [epochs]	50
Learning rate	0.008	End learning rate	10%
Learning rate decay [epochs]	40		

Chapter 6
Results and discussion

The following Section the summarizes the results obtained for each case study.

Also, the evaluation of the performance and the comparison between the state of the art, frequentist approach and MC Dropout for the C-MAPSS is addressed.

The structure follows the prior section, first, the regression tasks and then the classification. To validate our approach C-MAPSS is the first dataset to be analyzed. Then, Maryland Cracklines and GPM wind turbine. Finally, Ottawa and Torino bearing datasets shows another application to our development.

6.1 Regression tasks

6.1.1 NASA: C-MAPSS

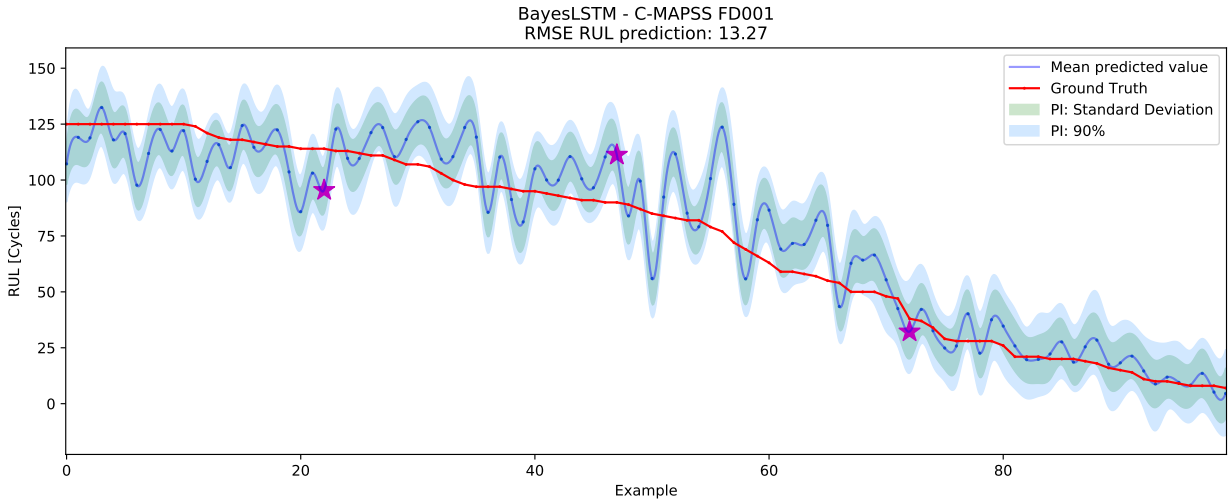
First, a summary of the results is presented for our approach as a mean of the performance of 3 runs. As clarified in Section 4.3, the mean RMSE corresponds to a large amount of single draws of the distribution and the mean of the RMSE for each draw with its respective standard deviation, the global RMSE (RMSE*) corresponds to the mean of the amount of single draws for each datapoint, generating the most likely distribution for each example. The last one is the comparable metric since it shows the overall performance of the regressor.

Table 6.1: Bayesian Recurrent Neural Networks results for C-MAPSS dataset. Mean of 3 runs.

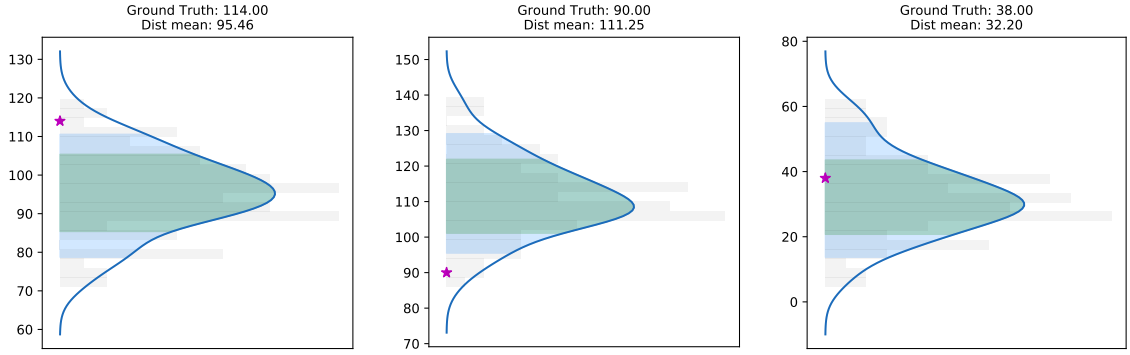
Dataset	FD001		FD002		FD003		FD004	
Model	Bayes LSTM	Bayes GRU	Bayes LSTM	Bayes GRU	Bayes LSTM	Bayes GRU	Bayes LSTM	Bayes GRU
Mean RMSE	17.73	17.82	23.97	24.07	17.36	16.33	27.28	27.74
STD	0.92	1.05	1.01	1.03	1.00	0.93	1.07	1.13
RMSE*	14.00	14.08	17.43	17.58	13.70	12.31	21.00	21.42

Also, the best result for each of the C-MAPSS dataset are presented. In this Figures, the comparison between each example point and its RUL is shown an the top figure an then, the distribution of 3 different examples (equispaced) are shown with the respective distributions.

The Table with all the results is at the Appendix B and plots for the other best runs are summarized at the Appendix C.

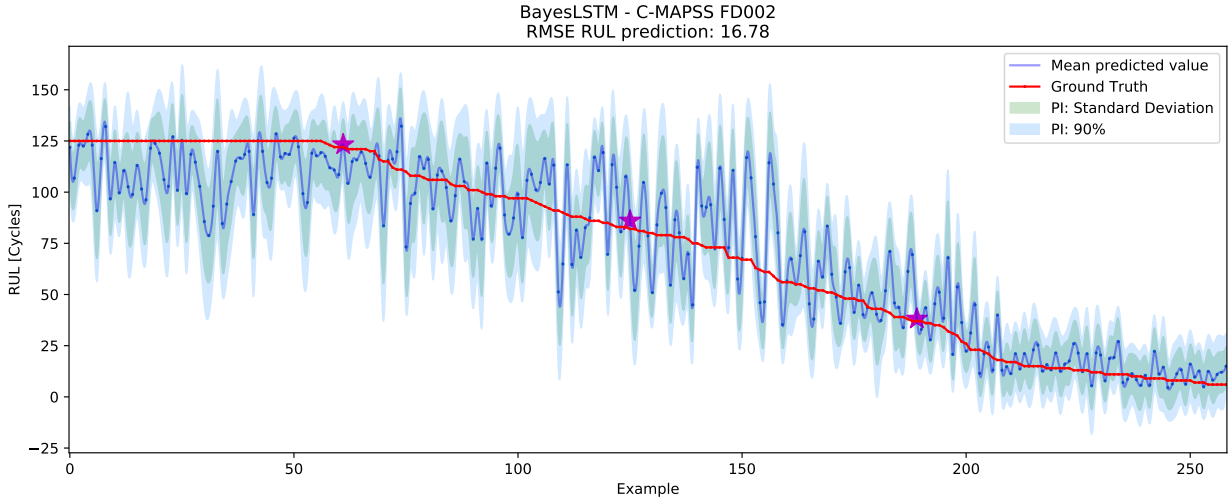


(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.

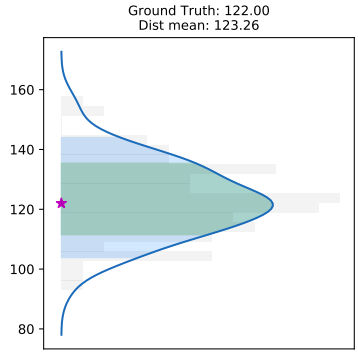


(b) 1rst star histogram (c) 2nd star histogram (d) 3rd star histogram

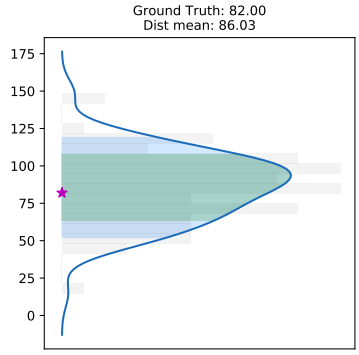
Figure 6.1: Best result for C-MAPSS FD001.



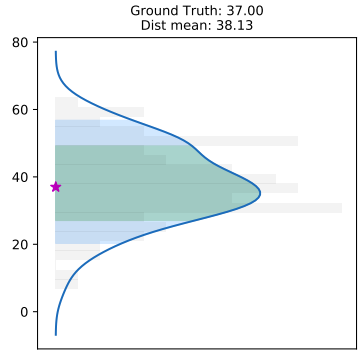
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram

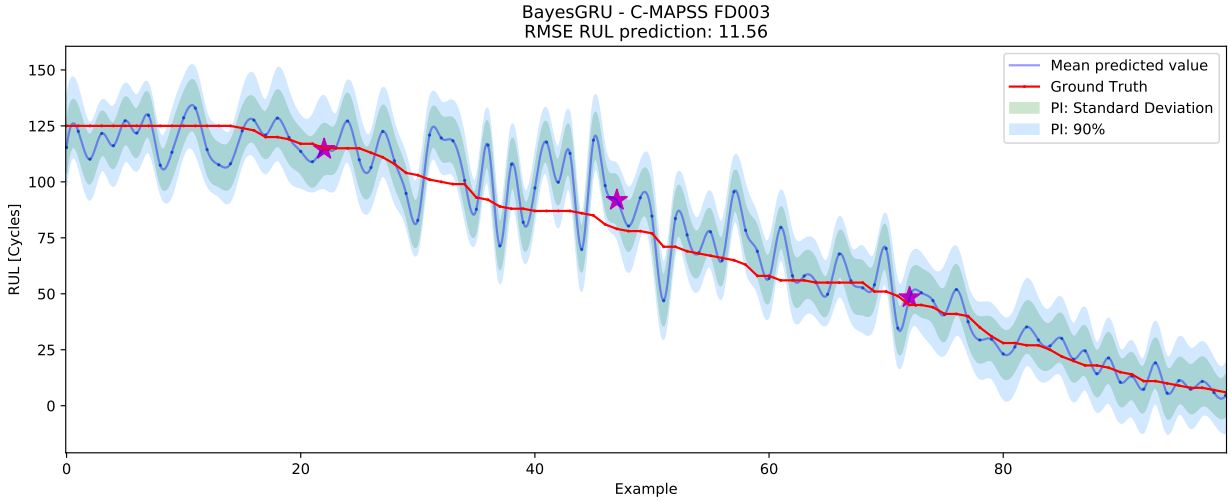


(c) 2nd star histogram

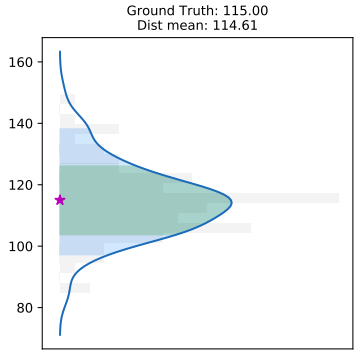


(d) 3rd star histogram

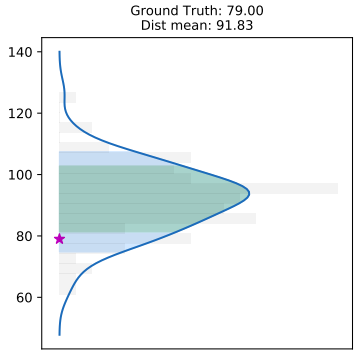
Figure 6.2: Best result for C-MAPSS FD002.



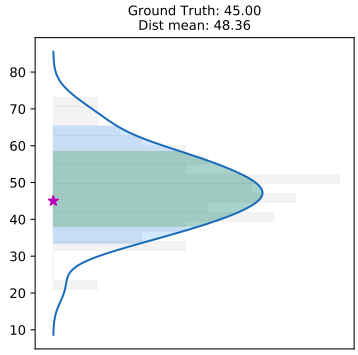
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram

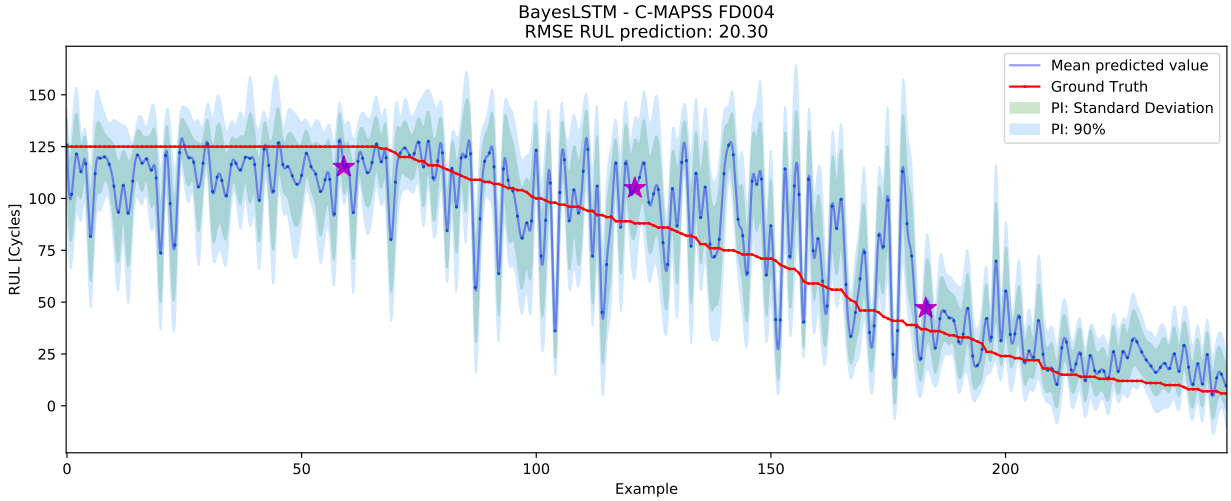


(c) 2nd star histogram

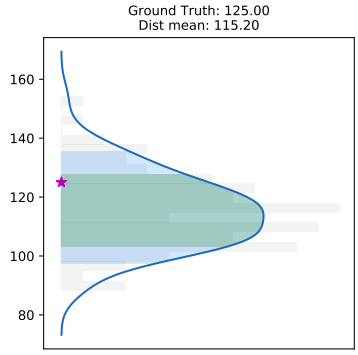


(d) 3rd star histogram

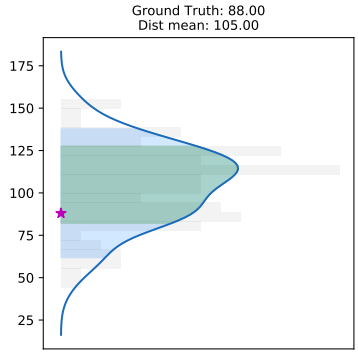
Figure 6.3: Best result for C-MAPSS FD003.



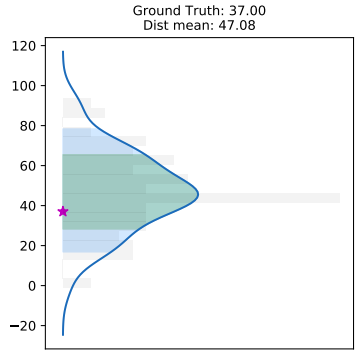
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram



(c) 2nd star histogram



(d) 3rd star histogram

Figure 6.4: Best result for C-MAPSS FD004.

Finally, a sample of the loss plot in a training of this architectures is shown.

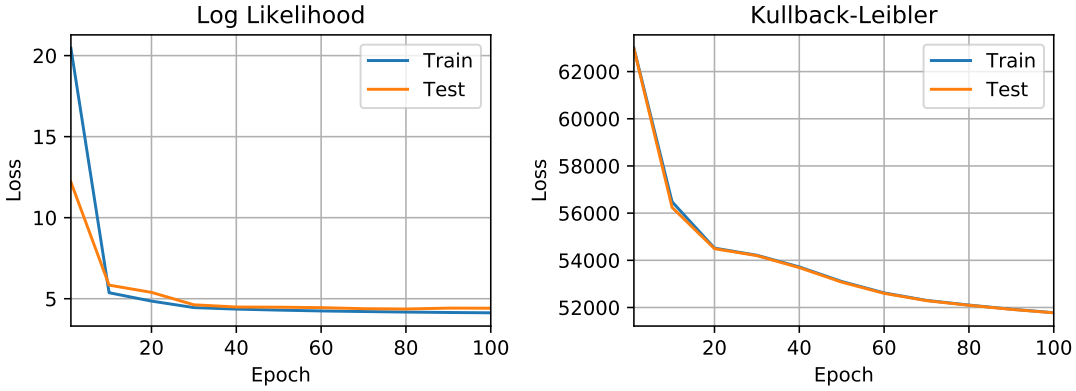


Figure 6.5: Example of a loss plot of C-MAPSS training.

The results shows good predictions, altough the plots seems to be noisy, the behavior is congruent with the training samples and the overall RMSE is satisfactory. Also, the loss plot shows good performance with decreasing values for both log likelihood and KL and without appreciable overfit. It is important to notice the difference of magnitude of these two values but as seen in Section 3.3 the KL acts only as a regularization term and the log likelihood is the main loss.

Then, the same architecture (and its hyperparameters) are trained with the frequentist approach. Training for the same amount of epochs at the frequentist model produces overfitted results, that leads to bad comparison between these two models, so the number of epochs is reduced to 20 to obtain unbiased results.

As it is expected, the frequentist models have nearly the half of parameters than the Bayesian approach, this is due to the definition of the Bayesian layers that is conformed with the mean weights and their perturbations. A breakdown of the number of weights per layer is in the Appendix D.

Table 6.2: Number of parameters in Bayesian and Frequentist approach.

	LSTM		GRU	
	Bayesian	Frequentist	Bayesian	Frequentist
FD001	74610	37313	71602	35809
FD002	56178	28097	53170	26593
FD003	74610	37313	71602	35809
FD004	50034	25025	47026	23521

The following Table 6.3 contains the mean results for 5 runs of the frequentist approach and the comparable result for the Bayesian approach:

Table 6.3: Comparison between average RMSE of Frequentist approach and Bayesian Recurrent Neural Networks

	LSTM		GRU	
Dataset	Frequentist Model RMSE	Bayesian Model RMSE*	Frequentist Model RMSE	Bayesian Model RMSE*
FD001	15.26	14.00	14.12	14.08
FD002	18.08	17.43	17.97	17.58
FD003	14.35	13.70	13.60	12.31
FD004	21.47	21.00	21.67	21.42

Since the Bayesian method have nearly double the parameters of the frequentist, the training phase of the Bayesian model takes longer. However, this longer training allow us to have a probability distribution as output, capturing the uncertainty and obtain better results. With this is possible to say that the first stages of the construction of a model could be tested with frequentist approach and then migrate the model to a Bayesian one to obtain the benefits listed above.

Then, the proposed approach is compared to MC Dropout as a Bayesian approximation. This method, consists in the application of dropout layers between dense layers and, unlike the normal application of dropout, applied the disconnection of random neurons in the evaluation process too, resulting in different outputs to a fixed input [18, 32, 42]. This methodology to address uncertainty has been questioned due to its result in simple tasks [43], its overconfident results [44] in classification and directly that does not quantify uncertainty [19]. However, still is an used method to deal with uncertainty.

As is another Bayesian approximation, it is possible to get the same metrics as in the propossed approach. The following Table 6.4 compares the best model for each dataset as a mean of 3 runs:

Table 6.4: Comparison between best average performance for each dataset between Bayesian and MC Dropout.

Dataset	FD001		FD002		FD003		FD004	
Model	Bayes LSTM	MC Dropout GRU	Bayes LSTM	MC Dropout LSTM	Bayes GRU	MC Dropout GRU	Bayes LSTM	MC Dropout LSTM
Mean RMSE	17.73	15.67	23.97	18.84	16.33	15.63	27.28	21.88
STD	0.92	0.68	1.01	0.33	0.93	0.65	1.07	0.27
RMSE*	14.00	14.25	17.43	18.24	12.31	14.52	21.00	21.53

It is noticeable that MC dropout have better mean RMSE than the proposed approach but in the global performance RMSE*, the Bayesian method implemented it is better, leading to better overall results. With this, our approach have better results.

Also, it is possible to compare the proposed approach best average performance for each dataset with some of the state of the art results:

Table 6.5: Proposed approach results compared with state of the art performance.

Model	MODBNE [12]	DCNN [45]	DLSTM [46]	AdaBN-CNN [47]	Proposed approach
Data	RMSE	RMSE	RMSE	RMSE	RMSE*
FD001	15.40	12.61	16.14	13.17	14.00
FD002	25.05	22.36	24.49	20.87	17.43
FD003	12.51	12.64	16.18	14.97	12.31
FD004	28.66	23.31	28.17	24.57	21.00

The models which are compared to our results are different types of architectures including: convolutional, recurrent, deep belief models and adaptative models. DCNN presents a standard deviation but is for 10 independent runs, not representing an uncertainty. It is important to notice that no one of the state of the art models address the uncertainty, giving a huge advantage of the proposed approach. Also, the Bayesian RNN outperforms the state of the art models considerably at the hardest datasets (FD002 and FD004) and the other two datasets have similar results than the other approaches.

6.1.2 Maryland: Crack Lines

As explained in the Section 5.1.2, this dataset have different runs that consists in different fatigue test samples. To correctly visualize their behavior, the regression plot stack all the runs and the independent plots are presented in the Appendix G.

Table 6.6: Bayesian Recurrent Neural Networks results for Maryland cracklines. Mean of 3 runs.

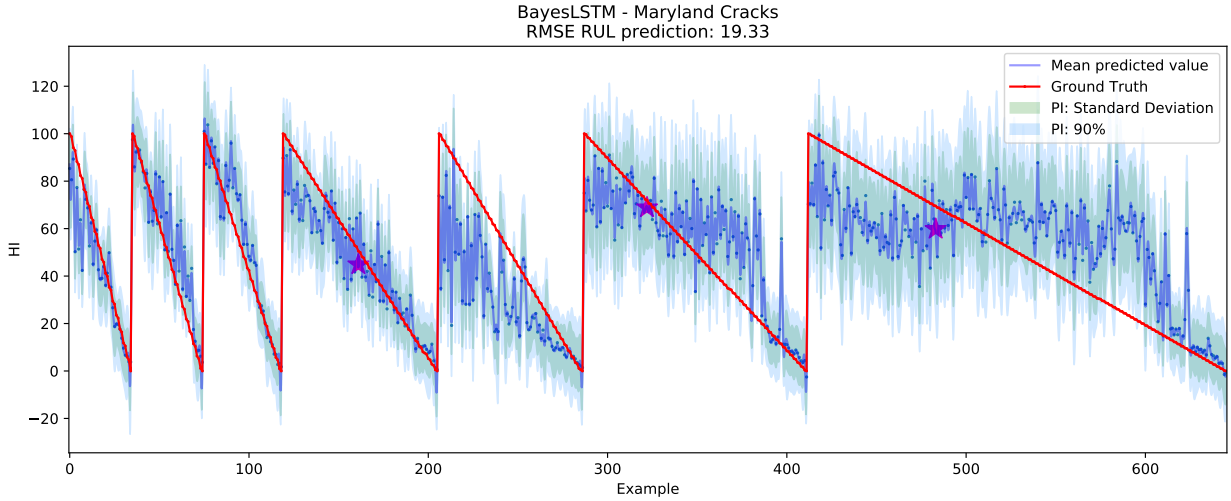
	Bayes LSTM	Bayes GRU
Mean RMSE	24.75	25.07
STD	0.59	0.57
Global RMSE	19.54	20.26

Table 6.7: Best result of Maryland dataset with BayesLSTM layer, per test set run.

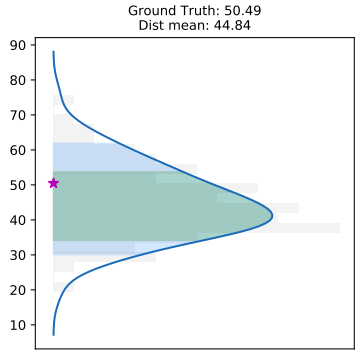
	Test run 1	Test run 2	Test run 3	Test run 4	Test run 5	Test run 6	Test run 7
Mearn RMSE	17.97	18.29	16.53	16.96	29.49	23.95	28.45
STD	1.67	1.96	1.50	1.20	1.70	1.34	1.13
RMSE*	14.16	13.05	10.80	11.76	26.15	17.98	22.18

Table 6.8: Best result of Maryland dataset with BayesGRU layer, per test set run.

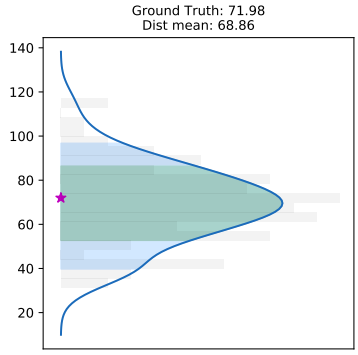
	Test run 1	Test run 2	Test run 3	Test run 4	Test run 5	Test run 6	Test run 7
Mearn RMSE	19.57	19.42	16.99	17.31	29.90	23.76	27.01
STD	1.86	1.87	1.89	1.19	1.30	1.63	1.06
RMSE*	15.53	14.38	10.65	11.55	27.01	16.97	20.68



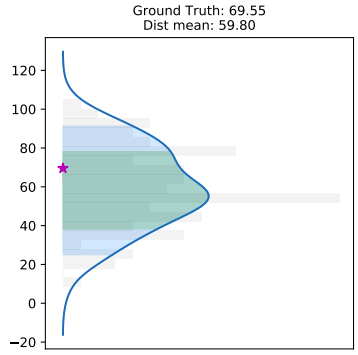
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram

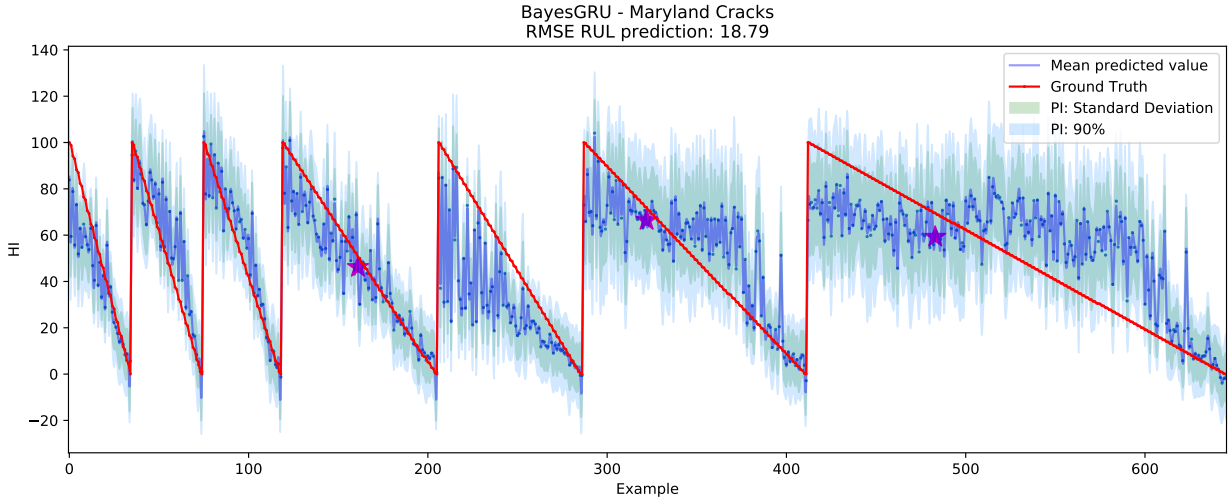


(c) 2nd star histogram

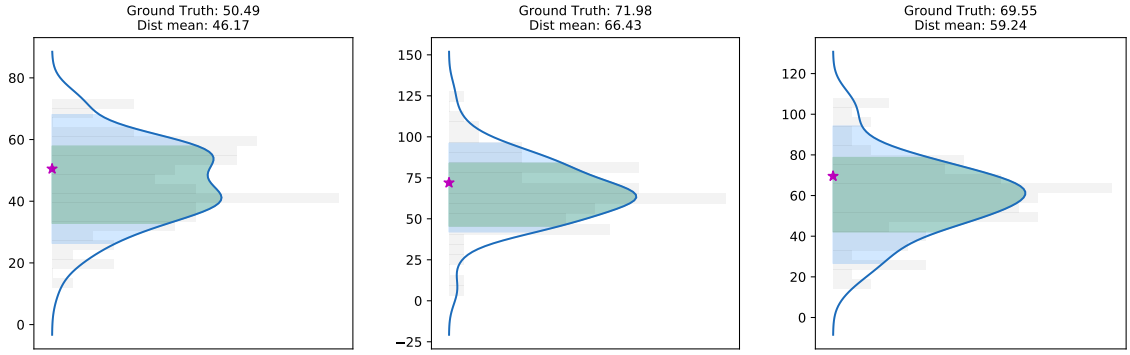


(d) 3rd star histogram

Figure 6.6: Best result for Maryland cracklines, Bayes LSTM.



(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram (c) 2nd star histogram (d) 3rd star histogram

Figure 6.7: Best result for Maryland cracklines, Bayes GRU.

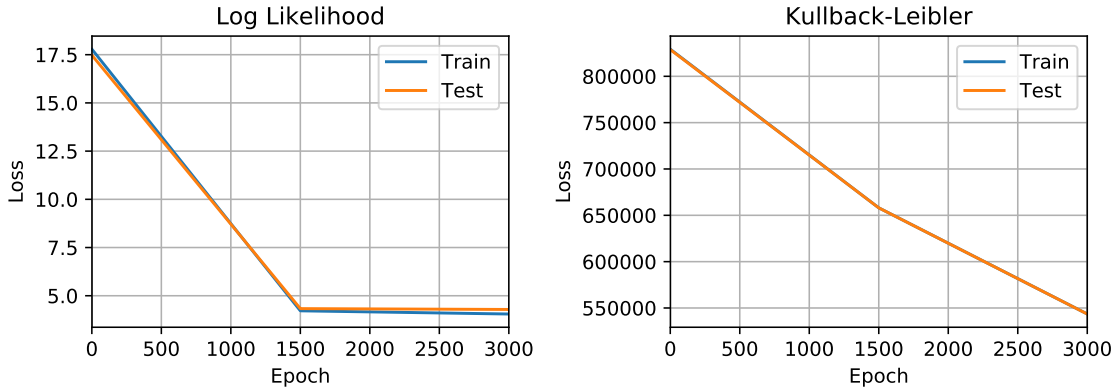


Figure 6.8: Example of a loss plot of Maryland dataset training.

It is noticeable that the one with shorter lives are the ones with better results. This is probably due to the fact that the proposed health indicator decays from the beginning of the test and this is not always true affecting the longest runs.

However, the results are good enough to obtain the behavior of the health indicator addressing the task successfully.

The training with this dataset is stopped at the stabilization of the log-likelihood and the KL is not growing (this considers stabilization or decreacing behavior).

A future work with this dataset could consider the identification of the starting point of damage (which is critical to the last two experiments), but to the scope of this Master Thesis, the results proves the functionality of the Bayesian RNN architectures successfully.

6.1.3 GPM Systems: Wind Turbine Data

As GPM systems are a HI regression task the results are visualized in the same scheme as C-MAPSS results.

The mean results are presented in the Table 6.9 and all the runs are reported at the Appendix H

This dataset is different to the C-MAPSS and Maryland crack dataset due to the conformation of the acquired data. The prior two datasets have multiple training data through start to end of the lifespan but this dataset have only one lifecycle.

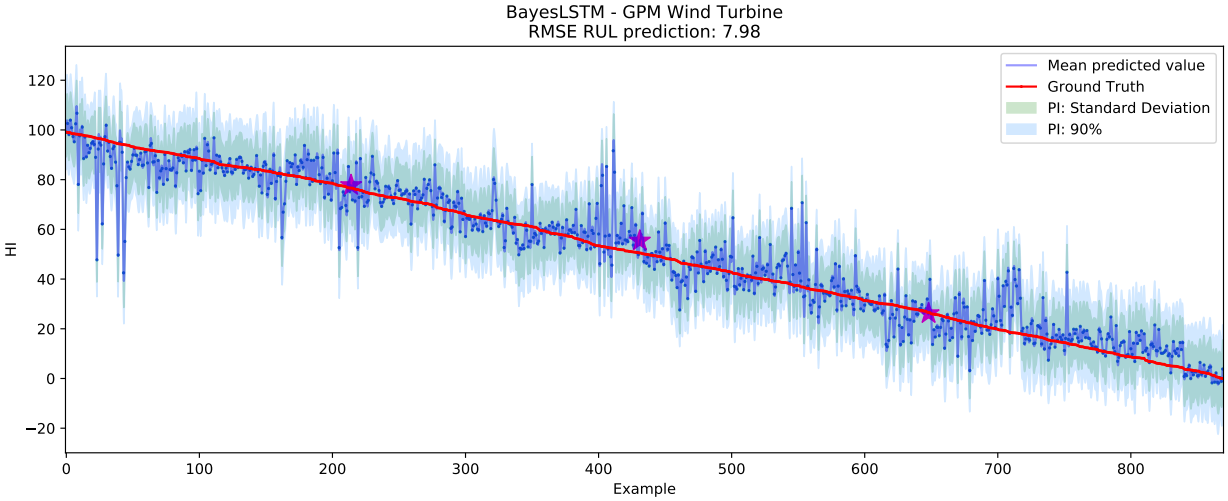
Also, the short measurements (6 seconds per day per 50 days) difficult the construction of a health indicator that represents the continuous wear and forces to split the dataset in train/test randomly.

With this, even if the results present a good behavior, this model is not tested in other runs so its impossible to predict their results in a real application (deployability).

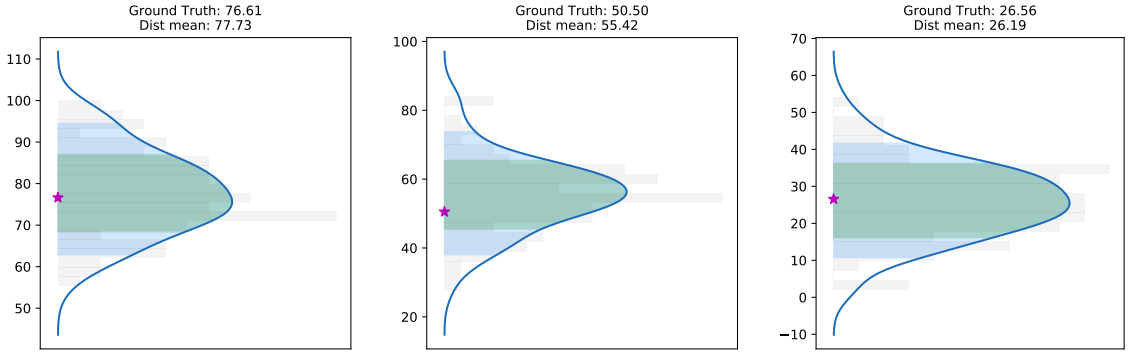
However, as Maryland crack lines dataset, this dataset confirms the utilization of the Bayesian RNNs as good representators of time-driven wears.

Table 6.9: Bayesian Recurrent Neural Networks results for Wind turbine. Mean of 3 runs.

	Bayes LSTM	Bayes GRU
Mean RMSE	13.40	13.71
STD	0.31	0.30
RMSE*	8.15	8.66

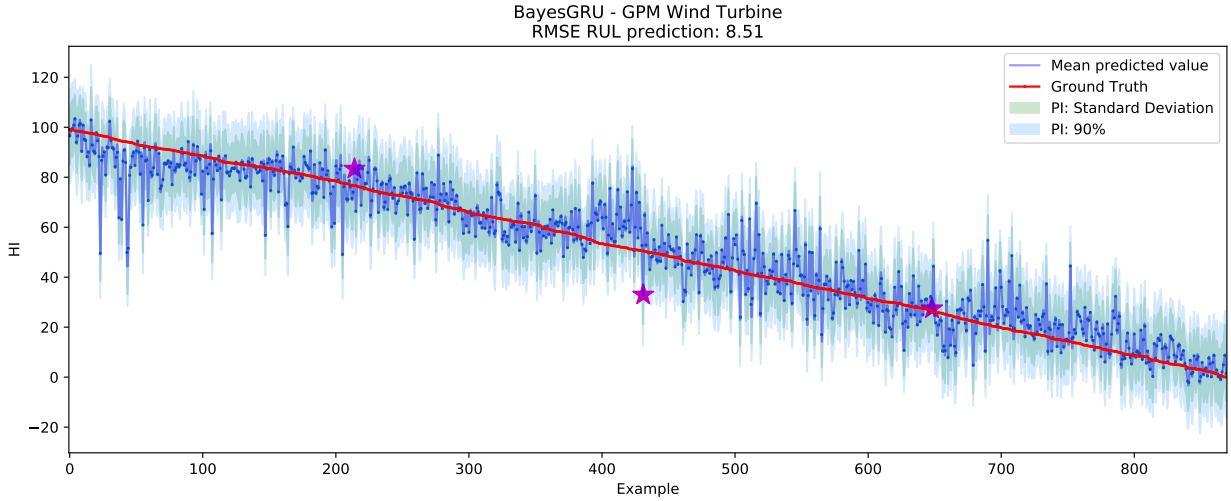


(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.

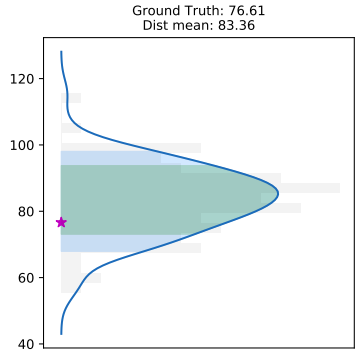


(b) 1rst star histogram (c) 2nd star histogram (d) 3rd star histogram

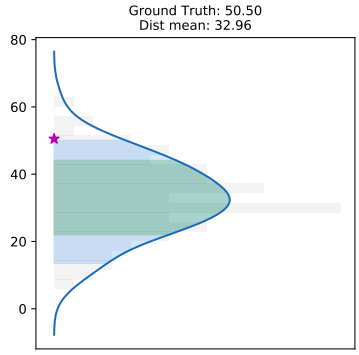
Figure 6.9: Best result for Wind Turbine Dataset. BayesLSTM architecture.



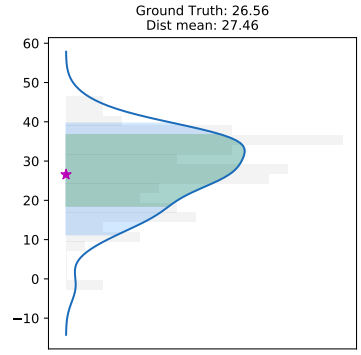
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram



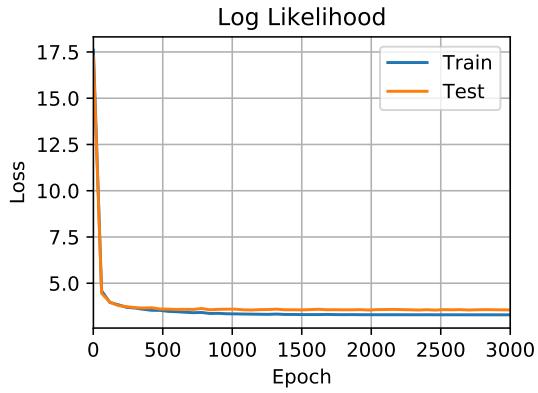
(c) 2nd star histogram



(d) 3rd star histogram

Figure 6.10: Best result for Wind Turbine Dataset. BayesGRU architecture.

Finally, a sample of the loss plot in a training of this architectures is shown.



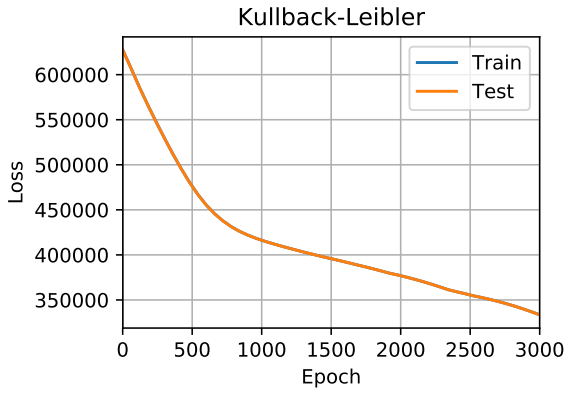


Figure 6.11: Example of a loss plot of Wind turbine training.

6.2 Classification tasks

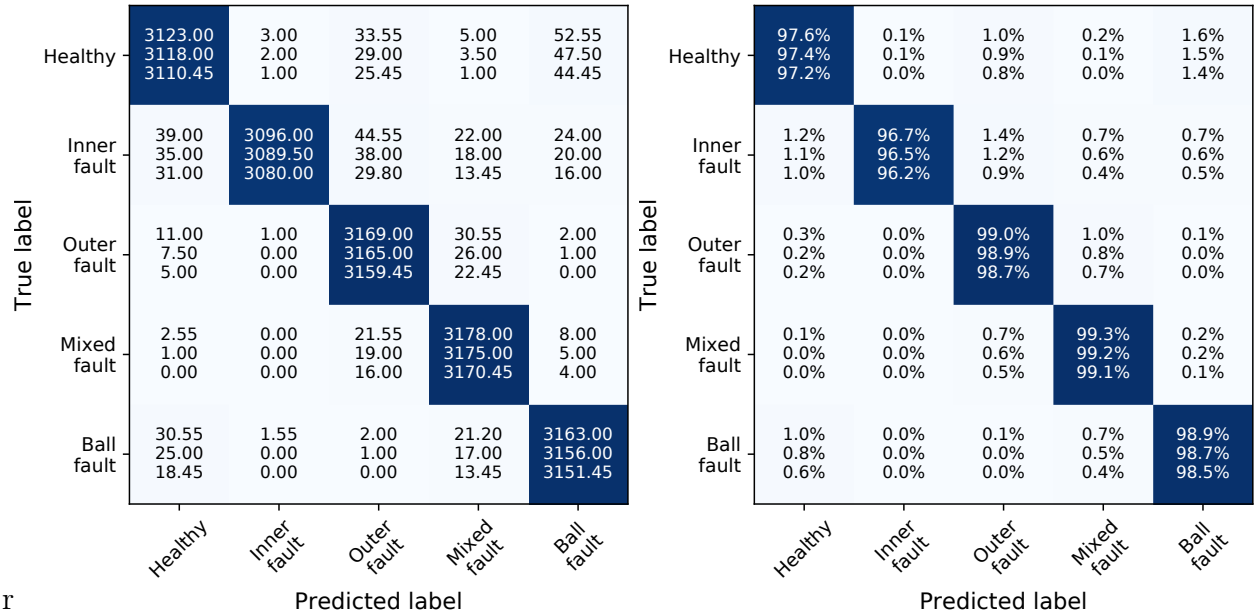
6.2.1 Ottawa: Bearing vibration Data

The results of this dataset are presented in the Table 6.10 as the summary of 3 runs per dataset.

Non-normalized and normalized confusion matrix are presented for the best result for this dataset at both Bayesian layers with a 90% probability interval.

Table 6.10: Bayesian Recurrent Neural Networks results for Ottawa bearing dataset. Mean of 3 runs.

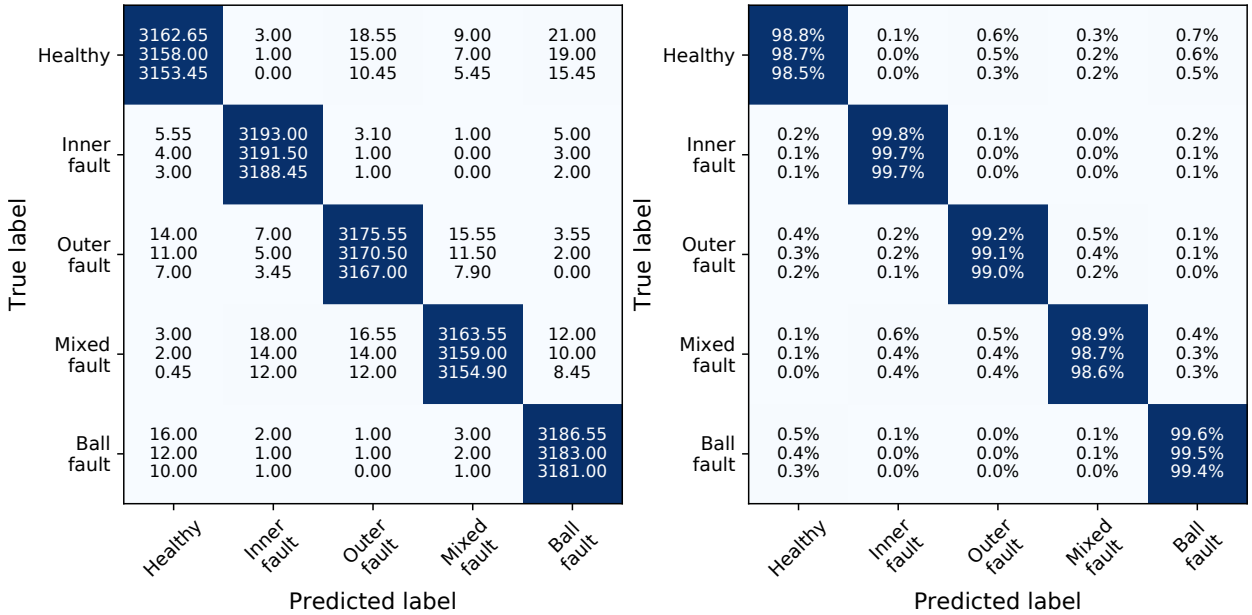
	Dataset 1		Dataset 2		Dataset 3	
	BayesLSTM	BayesGRU	BayesLSTM	BayesGRU	BayesLSTM	BayesGRU
Top 5%	93.68%	95.40%	98.97%	99.23%	97.51%	97.55%
50%	93.48%	95.21%	98.82%	99.09%	97.33%	97.37%
95%	93.26%	95.00%	98.67%	98.93%	97.13%	97.20%



r

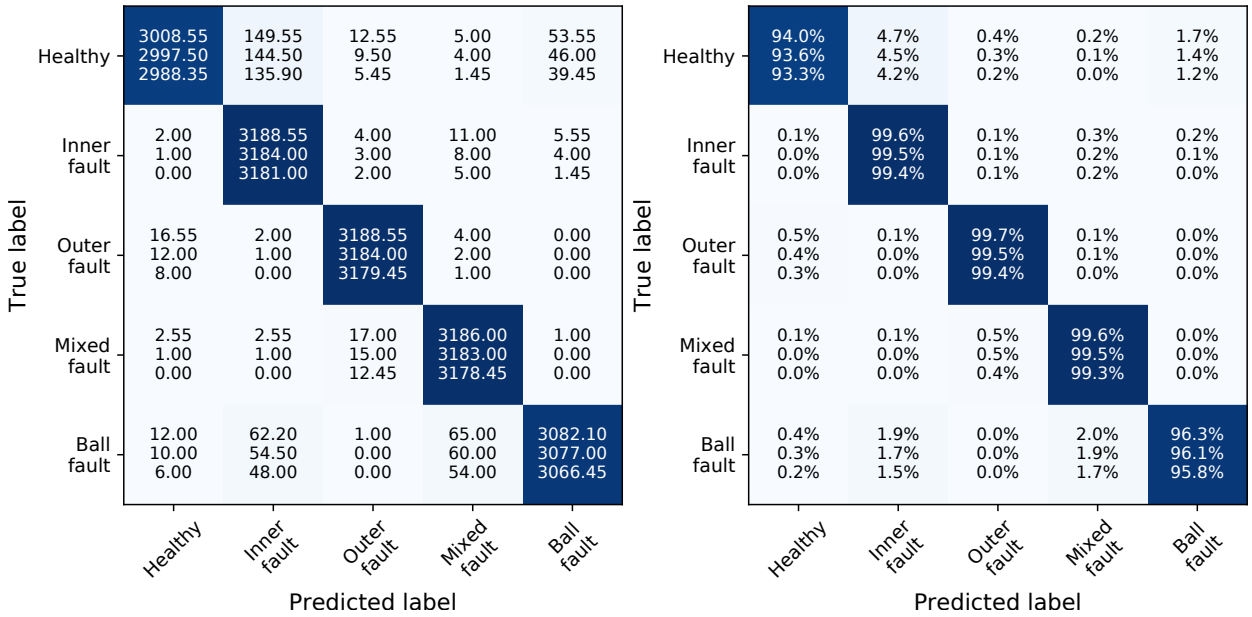
(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure 6.12: Best result for Ottawa bearing dataset. Dataset 1.



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure 6.13: Best result for Ottawa bearing dataset. Dataset 2.



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure 6.14: Best result for Ottawa bearing dataset. Dataset 3.

All the best confusion matrix corresponds to Bayesian GRUs cells. The Tables with the full runs per dataset can be seen in the Appendix I.

The classification task with this dataset shows very good results without a biased class (class with considerably lower results than the others). This is important in this dataset because the mixed fault class that consists in many different damages (Inner, outer and ball fault) and this could be a problematic class for the classifier, but the results presents robust outcomes. Even so, the best result for the Dataset 1 with Bayesian LSTM cell presents the above mistake with the Inner and Mixed fault.

As the Table 6.10 shows, the worst results obtained are with the first split of the dataset (5% less accuracy for the Bayesian LSTM and 4% for the Bayesian GRU). However, the accuracy is higher than a 90% being this very low probability of failure. Also, the split of the dataset are file-driven, so the contamination in the train-test split is negligible. This confirms that the model is extensible to new acquired data, which is the main issue with recurrent neural networks.

Also, the confusion matrix for the Bayesian LSTM models are presented at the end of the Appendix I.

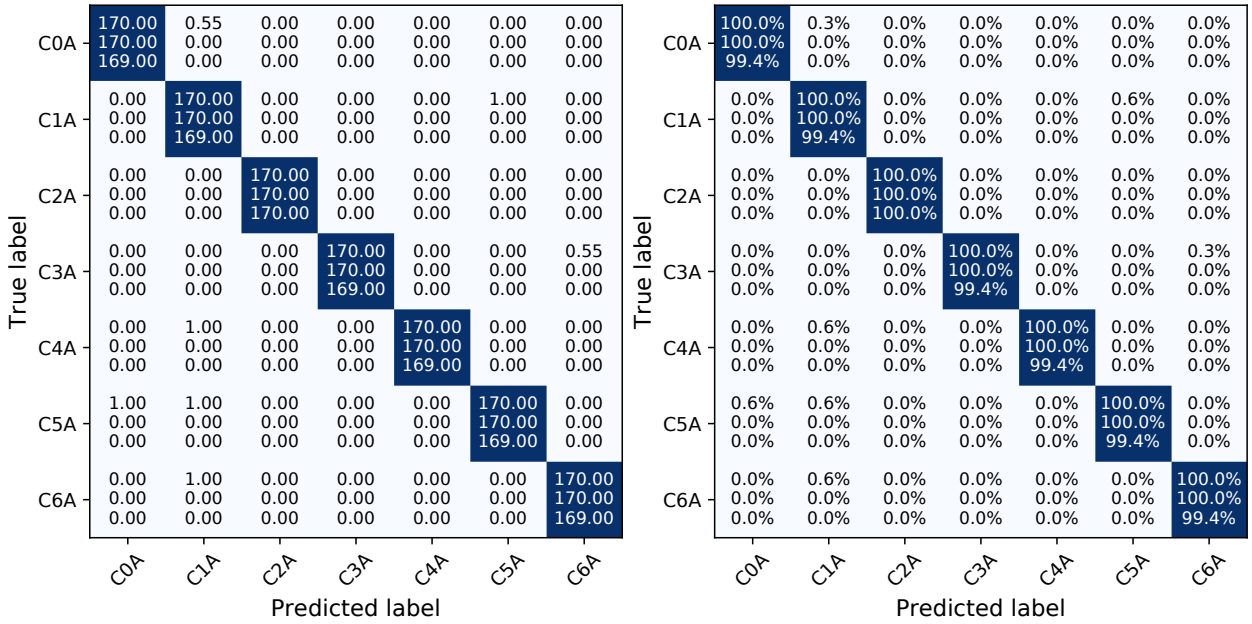
6.2.2 Politecnico di Torino: Rolling bearing Data

On the other hand, this dataset is not file-driven splitted since the construction of the dataset, even so, it is possible to split it by time which is explained at the Section 5.2.2.2. With this, the train-test contamination is negligible.

The above is really important in this dataset due the outstanding results that the proposed approach achieves. The Table 6.11 presents the mean results of 3 runs and the **worst** confusion matrix results for each Bayesian layer. The Table with all the results can be seen at the Appendix J.

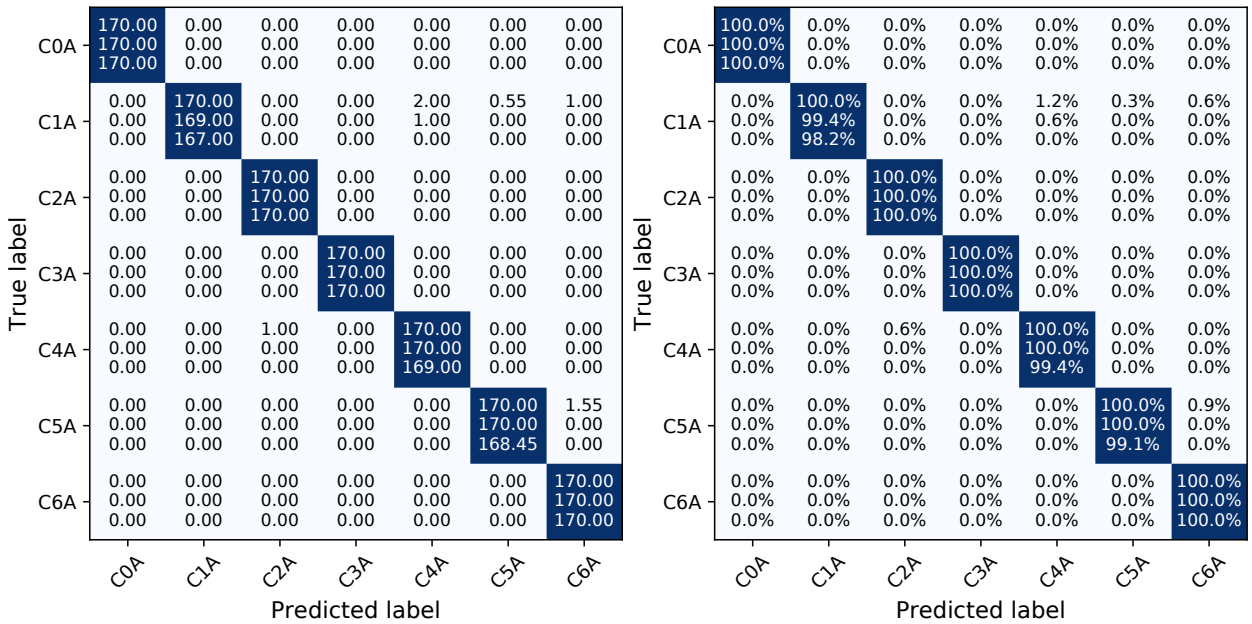
Table 6.11: Mean of 3 runs, Torino bearing dataset.

	BayesLSTM	BayesGRU
Top 5%	100.00%	100.00%
50%	100.00%	99.97%
95%	99.68%	99.79%



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure 6.15: Worst result Torino bearing dataset with Bayes LSTM layer.



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure 6.16: Worst result Torino bearing dataset with Bayes GRU layer.

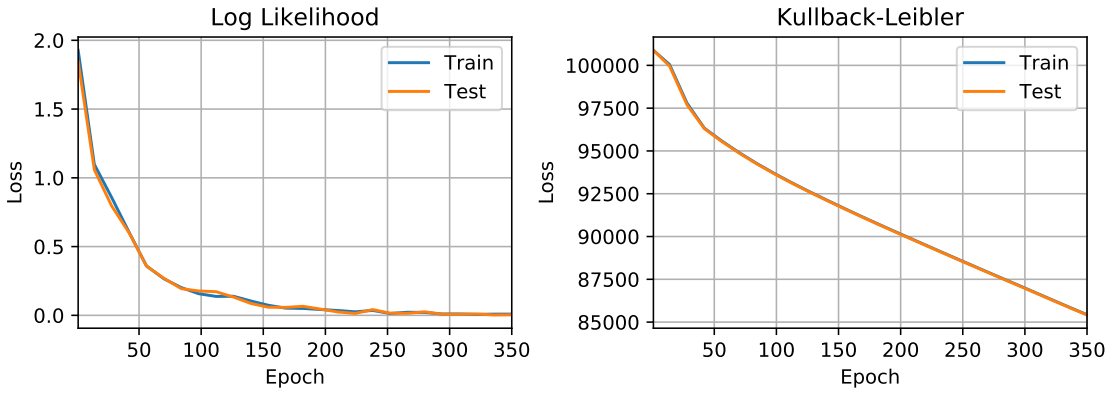


Figure 6.17: Example of a loss plot of Torino bearing dataset.

As can be seen the results obtained at this dataset are extremely good. It is possible to consider this as an overfitting but due to the train-test split and the behavior of the loss plot this is discarded.

With this, the health state is identified whatsoever amount of torque applied to the shaft.

Chapter 7
Concluding remarks and Future work

7.1 Concluding remarks

This thesis deals successfully with the construction of Bayesian Recurrent Neural Networks and proves its performance with regression tasks (RUL and HI prognosis) and classification tasks (health state classification) in complex mechanical systems based onto their sensory data.

The proposed Bayesian RNN method are implemented for LSTM and GRU layers, converting their inner gates to Bayesian dense flipout layers. The above produces easy to use keras-like layers that can be attached to every other keras layer with no effort. However, the use of this layers changes the training from a loss between the expected result and the predicted result (i.e. RMSE or cross entropy in regression and classification tasks) to a loss that is focused on the likelihood (data dependant) and complexity (prior dependant) since every layer uses a distribution as weights. The above allows the network to output a distribution that embody the uncertainty, which is a major advantage over frequentist approaches.

This is validated with C-MAPSS dataset, where outperforms other models like MODBNE, DCNN, DLSTM, AdaBN-CNN which are the most complex models with state of the art performance. Also, the result is compared to the frequentist approach of the same architecture which shows that the Bayesian approach not only is capable to address the uncertainty but also obtains better results at the cost of nearly double trainable parameters and longest training step. At the same dataset, MC Dropout is implemented as another approach to address the uncertainty. Again, the proposed approach shows better performance.

Two other regression tasks and two classification tasks are implemented with good results showing that the Bayesian recurrent neural networks can deal with sequential data to solve some of the common PHM problems. Also, the implementation in different datasets presents the extendability of the constructed Bayesian layers, allowing to be used in different areas, even outside PHM context, where the quantification of the uncertainty is an important problem.

7.2 Future work

The proposed Bayesian Recurrent Neural Networks are presented as a functional alternative to the frequentist RNNs that work with sequential data with the improvement that also quantify the uncertainty. However, the validations only uses Gaussian distribution as a prior, this could be not useful in every application but have shown good performance in the presented case studies. A future work could address other kind of problems like counting process (which can model renewal) or just try other priors with sensory data to evaluate the difference between the performance.

Also, as the construction of the layers does not involve necessary the application in the PHM context and only requires sequential data, another future work is apply the proposed framework into another context, as sequential modeling.

In the last years, attention models [48, 49, 50] has appeared as a new type of sequential models that have shown interesting applications in natural language processing which is a sequential type of data. However, this models uses a large amount of parameters (i.e. Facebook blender AI [50] have 90 millons of parameters at their smallest approach and 9.4 billions of parameters at their largest model). Since the application of the proposed approach nearly doubles the parameters of a network, the develop of Bayesian attention models could be an interesting challenge.

Bibliography

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

[2] L. Saidi, J. B. Ali, E. Bechhoefer, and M. Benbouzid, "Wind turbine high-speed shaft bearings health prognosis through a spectral kurtosis-derived indices and SVR," *Applied Acoustics*, vol. 120, pp. 1–8, May 2017. [Online]. Available: https://doi.org/10.1016/j.apacoust.2017.01.005

[3] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," pp. 799–834, may 2018.

[4] D. Siegel, H. Al-Atat, V. Shauche, L. Liao, J. Snyder, and J. Lee, "Novel method for rolling element bearing health assessment - A tachometer-less synchronously averaged envelope feature extraction technique," in *Mechanical Systems and Signal Processing*, vol. 29. Academic Press, may 2012, pp. 362–376.

[5] D. Wang and K. L. Tsui, "Statistical modeling of bearing degradation signals," *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1331–1344, dec 2017.

[6] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: A review," pp. 33–47, aug 2018.

[7] 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*, vol. 101, pp. 197–210, feb 2018.

[8] K. Goebel, B. Saha, A. Saxena, J. R. Celaya, and J. P. Christophersen, "Prognostics in battery health management," *IEEE Instrumentation and Measurement Magazine*, vol. 11, no. 4, pp. 33–40, 2008.

[9] H. Meng and Y. F. Li, "A review on prognostics and health management (PHM) methods of lithium-ion batteries," p. 109405, dec 2019.

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

[11] D. B. Verstraete, E. L. Droguett, V. Meruane, M. Modarres, and A. Ferrada, "Deep semi-supervised generative adversarial fault diagnostics of rolling element bearings," *Structural Health Monitoring*, vol. 19, no. 2, pp. 390–411, mar 2020. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1475921719850576

[12] Y. Zhang, R. Xiong, H. He, and M. G. Pecht, "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 5695–5705, jul 2018.

[13] C. Correa-Jullian, J. M. Cardemil, E. López Droguett, and M. Behzad, "Assessment of Deep Learning techniques for Prognosis of solar thermal systems," *Renewable Energy*, vol. 145, pp. 2178–2191, jan 2020.

[14] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," pp. 213–237, jan 2019.

[15] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. 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*, vol. 72-73, pp. 303–315, may 2016.

[16] W. Peng, Z. S. Ye, and N. Chen, "Bayesian Deep-Learning-Based Health Prognostics Toward Prognostics Uncertainty," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2283–2293, mar 2020.

[17] J. Hron, A. G. d. G. Matthews, and Z. Ghahramani, "Variational Gaussian Dropout is not Bayesian," nov 2017. [Online]. Available: http://arxiv.org/abs/1711.02989

[18] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 2016, pp. 1050–1059. [Online]. Available: http://proceedings.mlr.press/v48/gal16.html

[19] I. Osband and G. Deepmind, "Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout," Tech. Rep., 2016.

[20] T. Pearce, N. Anastassacos, M. Zaki, and A. Neely, "Bayesian Inference with Anchored Ensembles of Neural Networks, and Application to Exploration in Reinforcement Learning," may 2018. [Online]. Available: http://arxiv.org/abs/1805.11324

[21] P. L. McDermott and C. K. Wikle, "Bayesian Recurrent Neural Network Models for Forecasting and Quantifying Uncertainty in Spatial-Temporal Data," *Entropy*, vol. 21, no. 2, nov 2017. [Online]. Available: http://arxiv.org/abs/1711.00636

[22] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian Recurrent Neural Networks," apr 2017. [Online]. Available: http://arxiv.org/abs/1704.02798

[23] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, mar 2018.

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

[25] A. Y. Ng, "Machine Learning - Coursera," 2017. [Online]. Available: https://es.coursera.org/learn/machine-learning

[26] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[27] G. Bohouta and V. Këpuska, "Comparing speech recognition systems (microsoft api, google api and cmu sphinx)," *Int. Journal of Engineering Research and Application*, vol. 2248-9622, pp. 20–24, 03 2017.

[28] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman, H. Abraham, A. Mehler, A. Sipperley, A. Pettinato, B. Seppelt, L. Angell, B. Mehler, and B. Reimer, "MIT autonomous

vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation," *CoRR*, vol. abs/1711.06976, 2017. [Online]. Available: http://arxiv.org/abs/1711.06976

[29] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[30] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015.

[31] A. B. Owen, *Monte Carlo theory, methods and examples.*, 2013.

[32] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," 2015.

[33] J. Baalis Coble, "Merging Data Sources to Predict Remaining Useful Life – An Automated Method to Identify Prognostic Parameters," University of Tennessee, Tech. Rep., 2010. [Online]. Available: https://trace.tennessee.edu/utk{_}graddiss/683

[34] H. Yun and M. Modarres, "Measures of entropy to characterize fatigue damage in metallic materials," *Entropy*, vol. 21, p. 804, 08 2019.

[35] M. Bryant, M. Khonsari, and F. Ling, "On the thermodynamics of degradation," *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 464, pp. 2001–2014, 08 2008.

[36] C. Basaran and S. Nie, "An irreversible thermodynamics theory for damage mechanics of solids," *International Journal of Damage Mechanics - INT J DAMAGE MECH*, vol. 13, pp. 205–223, 07 2004.

[37] H. Yun and M. Modarres, "Measures of entropy to characterize fatigue damage in metallic materials," *Entropy*, vol. 21, no. 8, p. 804, Aug. 2019. [Online]. Available: https://doi.org/10.3390/e21080804

[38] M. Modarres and M. Amiri, "An entropy-based damage characterization," *Entropy*, vol. 16, pp. 6434–6463, 12 2014.

[39] M. Amiri and M. M. Khonsari, "Nondestructive estimation of remaining fatigue life: Thermography technique," *Journal of Failure Analysis and Prevention*, vol. 12, no. 6, pp. 683–688, Aug. 2012. [Online]. Available: https://doi.org/10.1007/s11668-012-9607-8

[40] "Practice for conducting force controlled constant amplitude axial fatigue tests of metallic materials." [Online]. Available: https://doi.org/10.1520/e0466-15

[41] J. B. Ali, L. Saidi, S. Harrath, E. Bechhoefer, and M. Benbouzid, "Online automatic diagnosis of wind turbine bearings progressive degradations under real experimental conditions based on unsupervised machine learning," *Applied Acoustics*, vol. 132, pp. 167–181, Mar. 2018. [Online]. Available: https://doi.org/10.1016/j.apacoust.2017.11.021

[42] S. ichi Maeda, "A bayesian encourages dropout," 2014.

[43] A. Y. K. Foong, D. R. Burt, Y. Li, and R. E. Turner, "On the expressiveness of approximate inference in bayesian neural networks," 2019.

[44] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," 2016.

[45] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using

deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, Apr. 2018. [Online]. Available: https://doi.org/10.1016/j.ress.2017.11.021

[46] J. Wu, K. Hu, Y. Cheng, H. Zhu, X. Shao, and Y. Wang, "Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network," *ISA Transactions*, vol. 97, pp. 241–250, Feb. 2020. [Online]. Available: https://doi.org/10.1016/j.isatra.2019.07.004

[47] J. Li, X. Li, and D. He, "Domain adaptation remaining useful life prediction method based on AdaBN-DCNN," in *2019 Prognostics and System Health Management Conference (PHM-Qingdao)*. IEEE, Oct. 2019. [Online]. Available: https://doi.org/10.1109/phm-qingdao46334.2019.8942857

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[49] A. Galassi, M. Lippi, and P. Torroni, "Attention, please! A critical review of neural attention models in natural language processing," *CoRR*, vol. abs/1902.02181, 2019. [Online]. Available: http://arxiv.org/abs/1902.02181

[50] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, Y.-L. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," 2020.

Appendix

A Statistical parameters extracted from signals as features

- Max Value
- Root Mean Square (RMS)
- Peak to peak
- Crest Factor
- Mean
- Variance
- Skewness
- Kurtosis
- 5th moment
- The mean for each of the 5 first frequency bands of the Fourier frequency transform.

B Results of Bayesian Recurrent Neural Networks, C-MAPSS dataset.

Table B.1: Bayesian Recurrent Neural Network, FD001.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	18.12	17.17	17.90	17.95	17.69	17.83
STD	0.84	0.98	0.93	1.07	1.08	1.00
RMSE*	14.49	13.27	14.23	14.34	13.80	14.10

Table B.2: Bayesian Recurrent Neural Network, FD002.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	23.97	24.21	23.74	23.42	24.22	24.56
STD	1.02	1.00	1.01	0.92	1.05	1.10
RMSE*	17.47	18.03	16.78	17.37	17.32	18.05

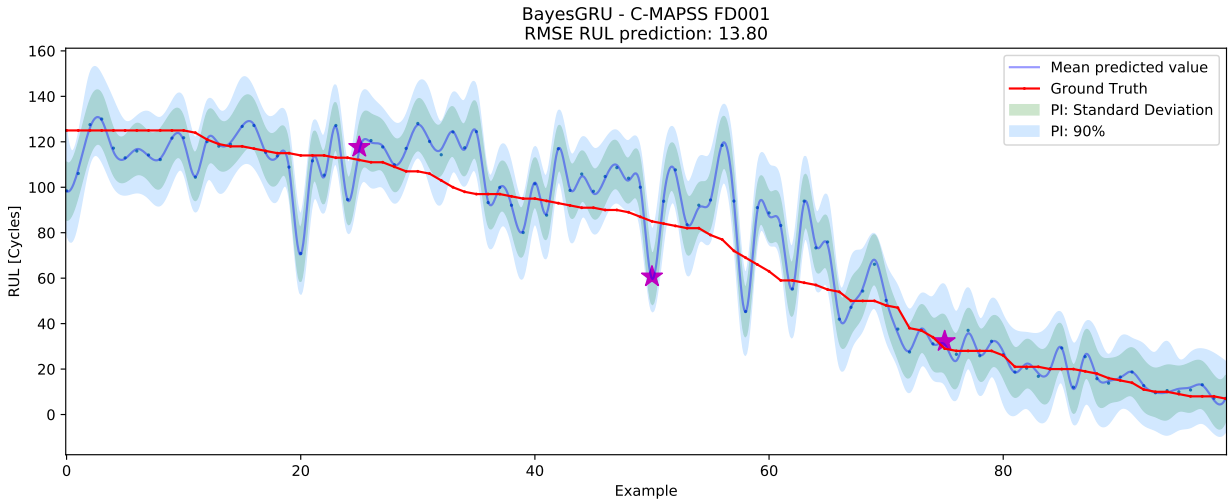
Table B.3: Bayesian Recurrent Neural Network, FD003.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	17.33	17.07	17.69	15.69	16.62	16.67
STD	1.05	0.93	1.03	0.88	0.99	0.91
RMSE*	13.56	13.35	14.20	11.56	12.66	12.71

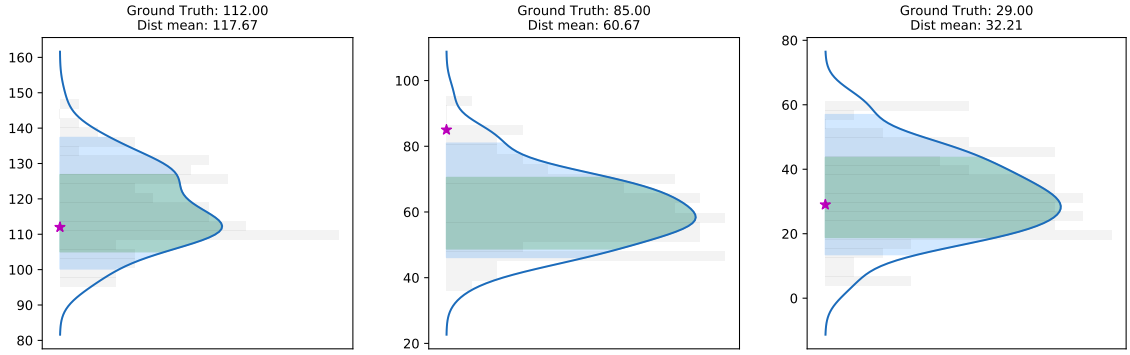
Table B.4: Bayesian Recurrent Neural Network, FD004.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	27.94	26.74	27.15	27.19	27.63	28.39
STD	0.96	1.02	1.22	1.12	1.10	1.17
RMSE*	22.05	20.30	20.66	20.80	21.38	22.07

C Plots of the results of Bayesian RNN, C-MAPSS.

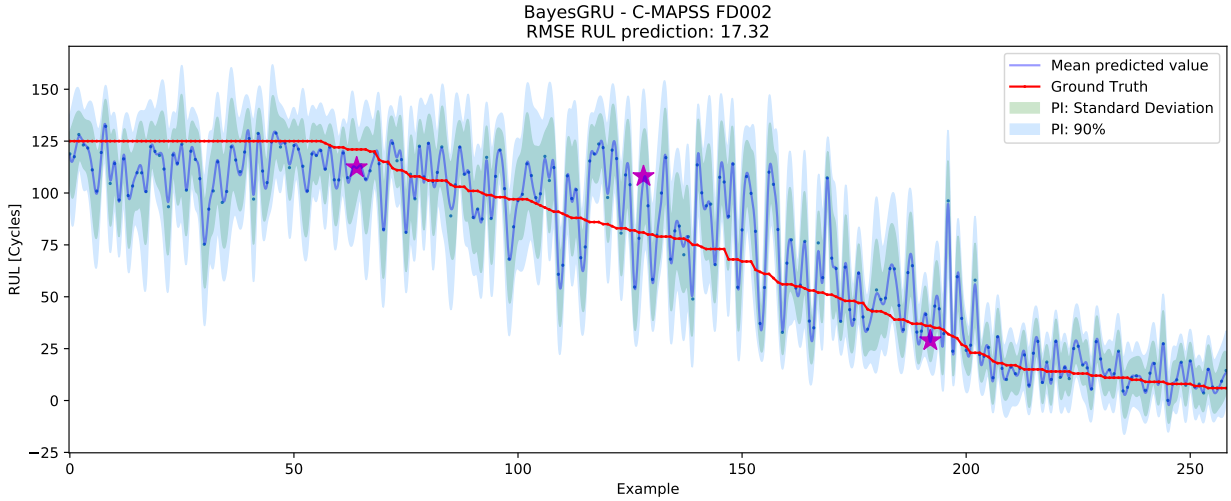


(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.

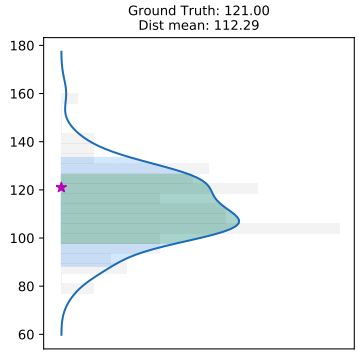


(b) 1rst star histogram (c) 2nd star histogram (d) 3rd star histogram

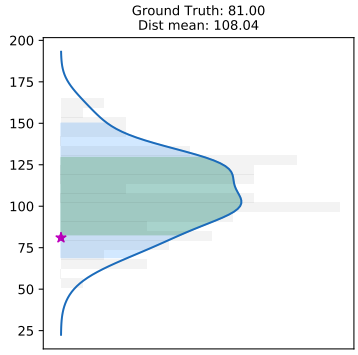
Figure C.1: Best result for C-MAPSS FD001 with Bayesian GRU cell.



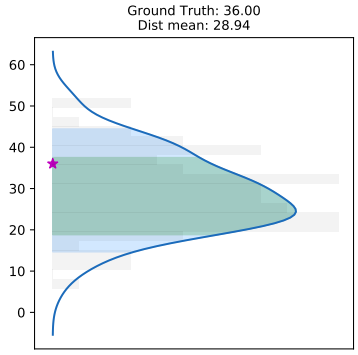
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram

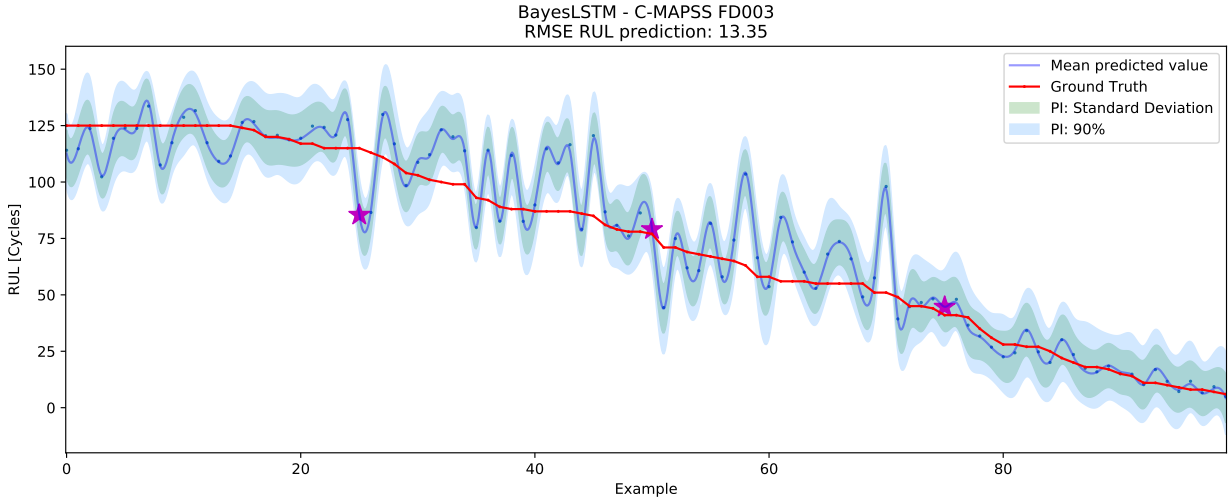


(c) 2nd star histogram

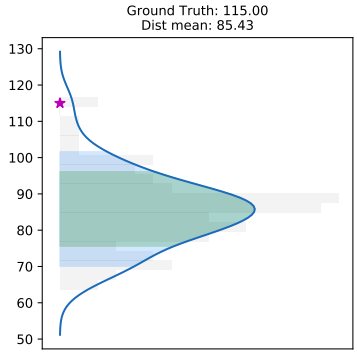


(d) 3rd star histogram

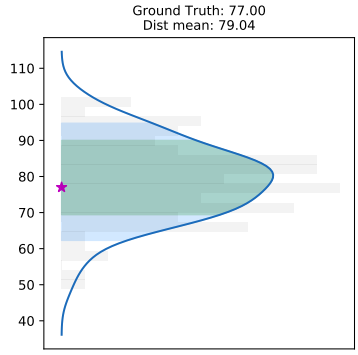
Figure C.2: Best result for C-MAPSS FD002 with Bayesian GRU cell.



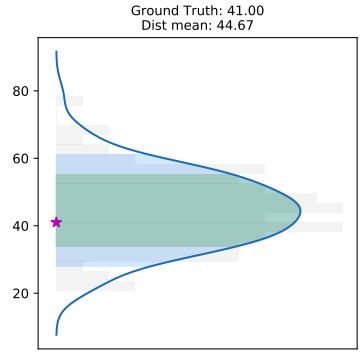
(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram

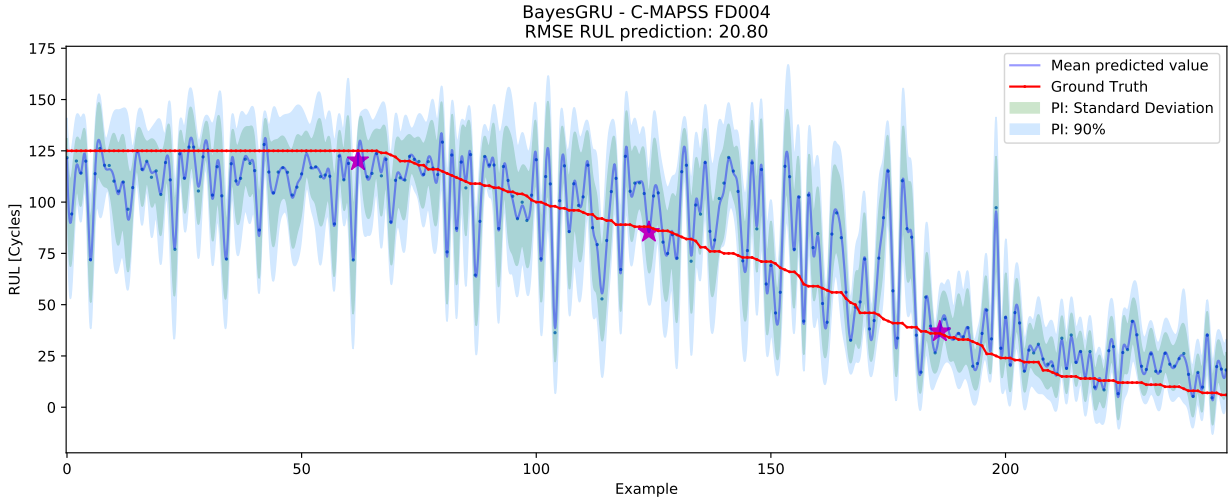


(c) 2nd star histogram

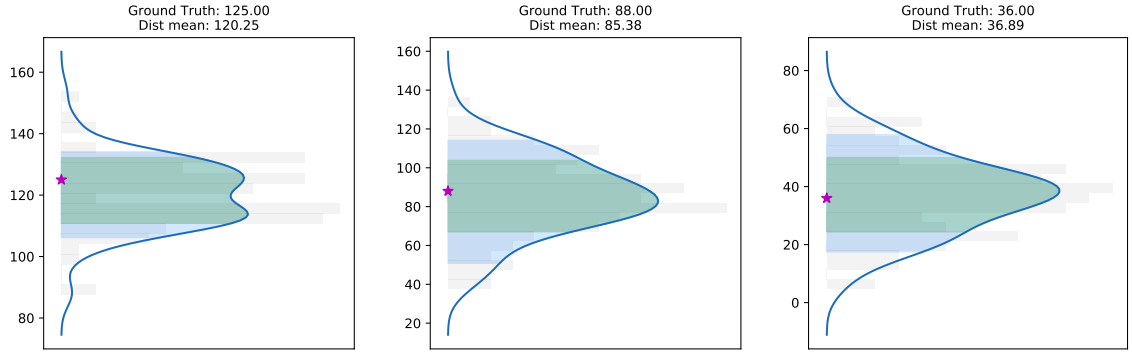


(d) 3rd star histogram

Figure C.3: Best result for C-MAPSS FD003 with Bayesian LSTM cell.



(a) RUL prediction with mean predicted value, 2 probability intervals and the ground truth.



(b) 1rst star histogram (c) 2nd star histogram (d) 3rd star histogram

Figure C.4: Best result for C-MAPSS FD004 with Bayesian GRU cell.

D Comparison tables of the number of trainable parameters in Frequentist and Bayesian models for C-MAPSS Dataset.

Table D.1: Parameter table, FD001.

Dataset	FD001	LSTM		GRU	
	Output shape	Bayesian	Frequentist	Bayesian	Frequentist
Input	(None, 30, 14)	0	0	0	0
LSTM	(None, 30, 32)	12032	6016	9024	4512
Flatten	(None, 960)	0	0	0	0
Dense 1	(None, 32)	61472	30752	61472	30752
Dense 2	(None, 16)	1040	528	1040	528
Output Bayesian	(None, 2)	66	-	66	-
Output Frequentist	(None, 1)	-	17	-	17
Total Parameters		74610	37313	71602	35809

Table D.2: Parameter table, FD002.

Dataset	FD002	LSTM		GRU	
	Output shape	Bayesian	Frequentist	Bayesian	Frequentist
Input	(None, 21, 14)	0	0	0	0
LSTM	(None, 21, 32)	12032	6016	9024	4512
Flatten	(None, 672)	0	0	0	0
Dense 1	(None, 32)	43040	21536	43040	21536
Dense 2	(None, 16)	1040	528	1040	528
Output Bayesian	(None, 2)	66	-	66	-
Output Frequentist	(None, 1)	-	17	-	17
Total Parameters		56178	28097	53170	26593

Table D.3: Parameter table, FD003.

Dataset	FD003	LSTM		GRU	
	Output shape	Bayesian	Frequentist	Bayesian	Frequentist
Input	(None, 30, 14)	0	0	0	0
LSTM	(None, 30, 32)	12032	6016	9024	4512
Flatten	(None, 960)	0	0	0	0
Dense 1	(None, 32)	61472	30752	61472	30752
Dense 2	(None, 16)	1040	528	1040	528
Output Bayesian	(None, 2)	66	-	66	-
Output Frequentist	(None, 1)	-	17	-	17
Total Parameters		74610	37313	71602	35809

Table D.4: Parameter table, FD004.

Dataset	FD004	LSTM		GRU	
	Output shape	Bayesian	Frequentist	Bayesian	Frequentist
Input	(None, 18, 14)	0	0	0	0
LSTM	(None, 18, 32)	12032	6016	9024	4512
Flatten	(None, 576)	0	0	0	0
Dense 1	(None, 32)	36896	18464	36896	18464
Dense 2	(None, 16)	1040	528	1040	528
Output Bayesian	(None, 2)	66	-	66	-
Output Frequentist	(None, 1)	-	17	-	17
Total Parameters		50034	25025	47026	23521

E Results of Frequentist Recurrent Neural networks, C-MAPSS dataset.

Table E.1: All runs of Frequentist Recurrent Neural Network approach.

Dataset	FD001		FD002		FD003		FD004	
Model	LSTM	GRU	LSTM	GRU	LSTM	GRU	LSTM	GRU
1	14.00	14.09	17.82	17.61	13.70	13.19	21.64	21.70
2	16.51	13.71	18.09	18.88	15.27	13.58	21.27	21.48
3	15.56	14.30	17.99	17.73	14.18	14.21	21.66	21.82
4	14.39	13.76	18.93	17.90	14.61	13.40	21.20	21.79
5	15.82	14.75	17.58	17.75	13.96	13.61	21.57	21.56
Mean	15.26	14.12	18.08	17.97	14.35	13.60	21.47	21.67

F Results of MC Dropout Recurrent Neural networks, C-MAPSS dataset.

Table F.1: Summary of MC Dropout Models, C-MAPSS dataset.

Dataset	FD001		FD002		FD003		FD004	
Model	MC Dropout LSTM	MC Dropout GRU	MC Dropout LSTM	MC Dropout GRU	MC Dropout LSTM	MC Dropout GRU	MC Dropout LSTM	MC Dropout GRU
Mean RMSE	16.19	15.67	18.84	19.09	16.02	15.63	21.88	22.19
STD	0.68	0.68	0.33	0.31	0.57	0.65	0.27	0.28
RMSE*	14.96	14.25	18.24	18.51	15.11	14.52	21.53	21.75

Table F.2: MC Dropout Recurrent Neural Network, FD001.

	MC Dropout LSTM			MC Dropout GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	17.23	16.17	15.17	15.32	16.19	15.48
STD	0.76	0.73	0.52	0.64	0.77	0.64
RMSE*	16.07	14.40	14.41	14.05	14.50	14.19

Table F.3: MC Dropout Recurrent Neural Network, FD002.

	MC Dropout LSTM			MC Dropout GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	19.44	18.79	18.28	18.83	19.19	19.25
STD	0.40	0.30	0.26	0.28	0.32	0.33
RMSE*	18.58	18.36	17.77	18.30	18.60	18.63

Table F.4: MC Dropout Recurrent Neural Network, FD003.

	MC Dropout LSTM			MC Dropout GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	15.35	16.12	16.59	14.81	14.96	17.13
STD	0.66	0.54	0.48	0.60	0.69	0.67
RMSE*	14.17	15.28	15.87	13.82	13.77	15.96

Table F.5: MC Dropout Recurrent Neural Network, FD004.

	MC Dropout LSTM			MC Dropout GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	22.11	21.91	21.62	22.18	22.03	22.37
STD	0.26	0.25	0.30	0.27	0.31	0.26
RMSE*	21.76	21.62	21.22	21.81	21.49	21.95

G Results of Bayesian Recurrent Neural networks, Maryland cracklines dataset.

Table G.1: Results of Bayesian Recurrent Neural networks, Maryland cracklines dataset.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	24.69	25.53	25.57	24.82	24.30	24.50
STD	0.57	0.64	0.57	0.62	0.56	0.58
RMSE*	19.33	20.31	20.14	18.85	18.79	19.17

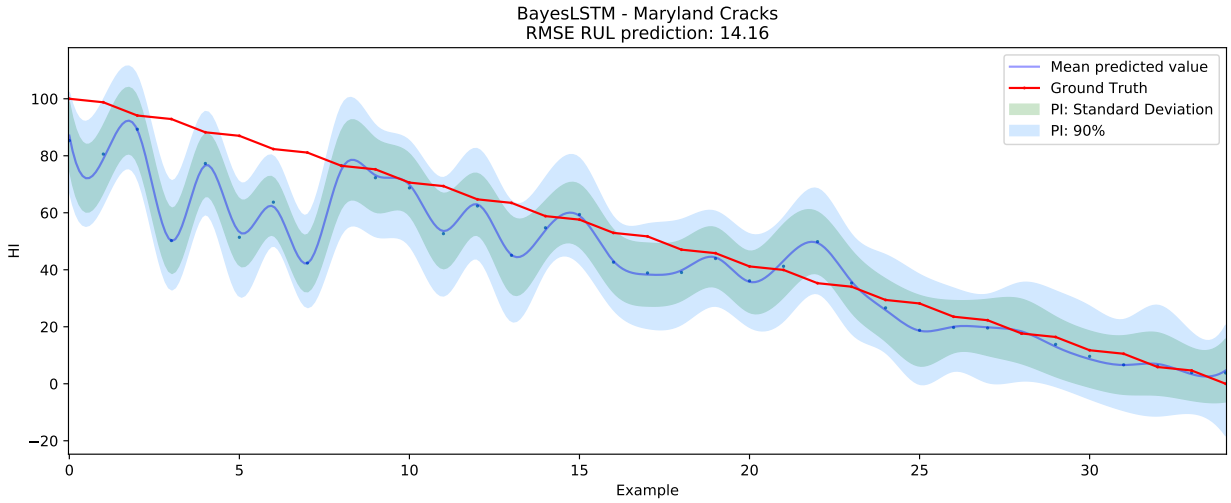


Figure G.1: First complete life cycle. Maryland cracklines, Bayesian LSTM.

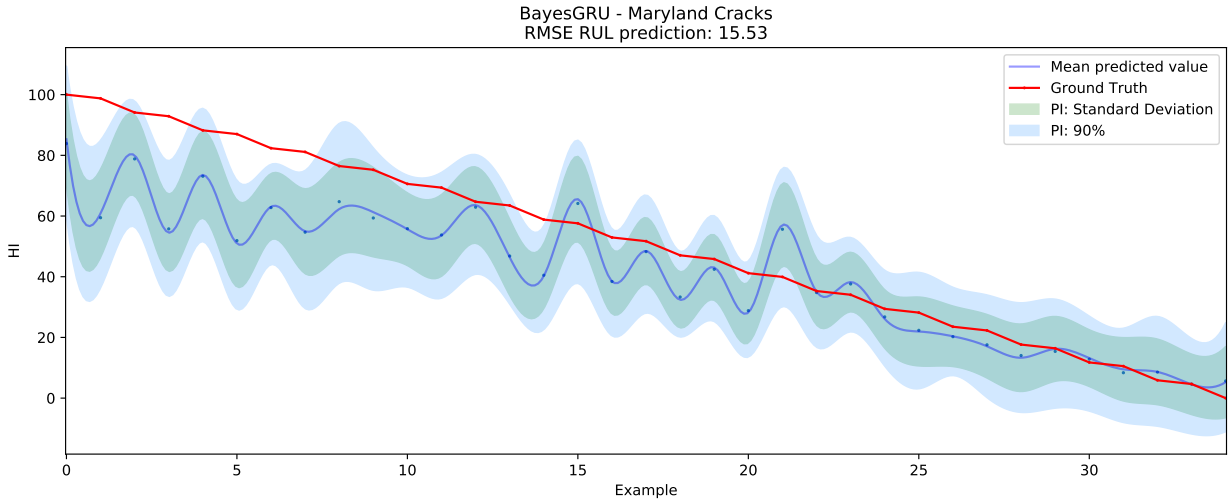


Figure G.2: First complete life cycle. Maryland cracklines, Bayesian GRU.

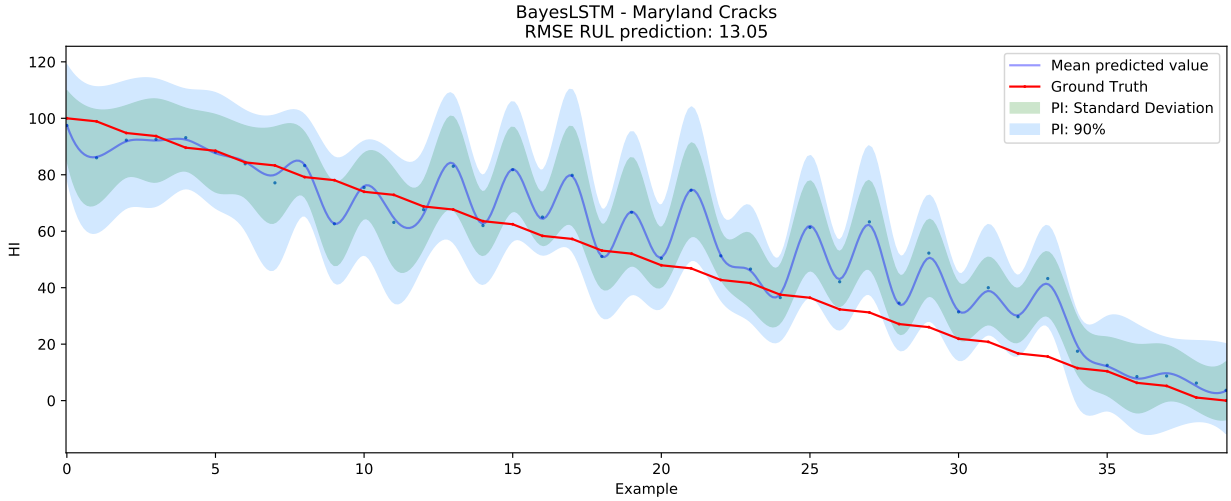


Figure G.3: Second complete life cycle. Maryland cracklines, Bayesian LSTM.

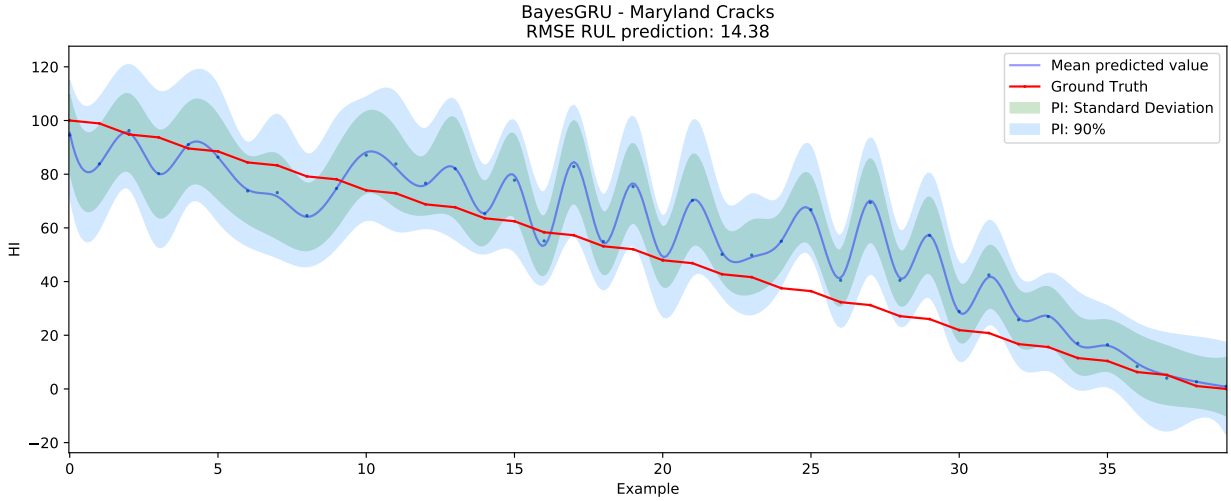


Figure G.4: Second complete life cycle. Maryland cracklines, Bayesian GRU.

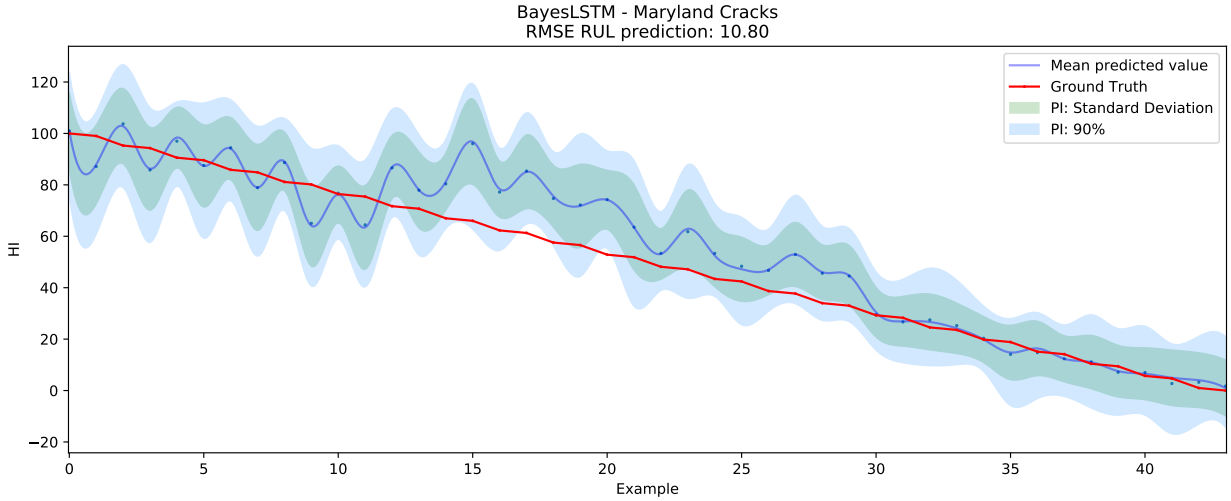


Figure G.5: Third complete life cycle. Maryland cracklines, Bayesian LSTM.

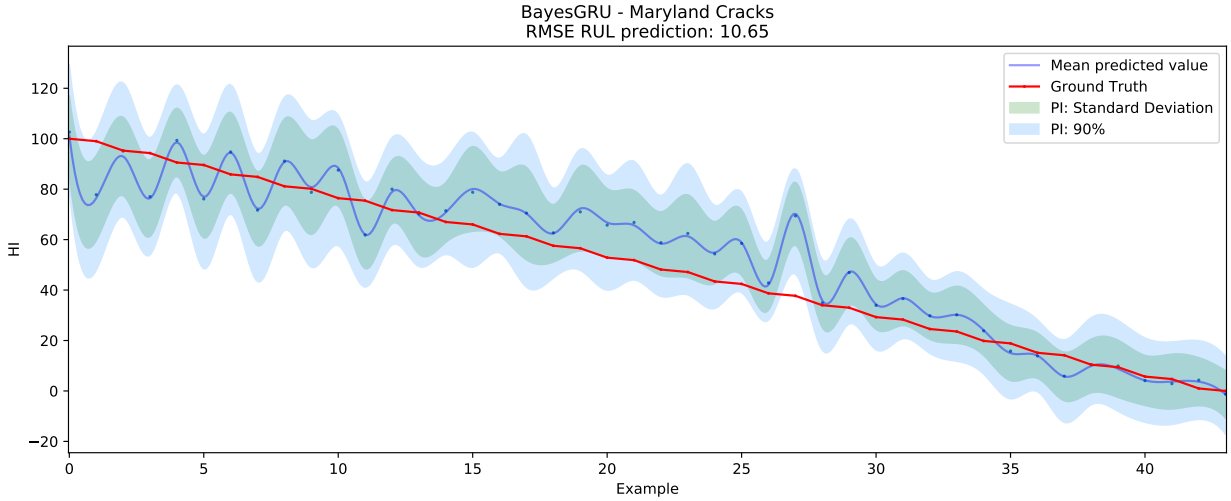


Figure G.6: Third complete life cycle. Maryland cracklines, Bayesian GRU.

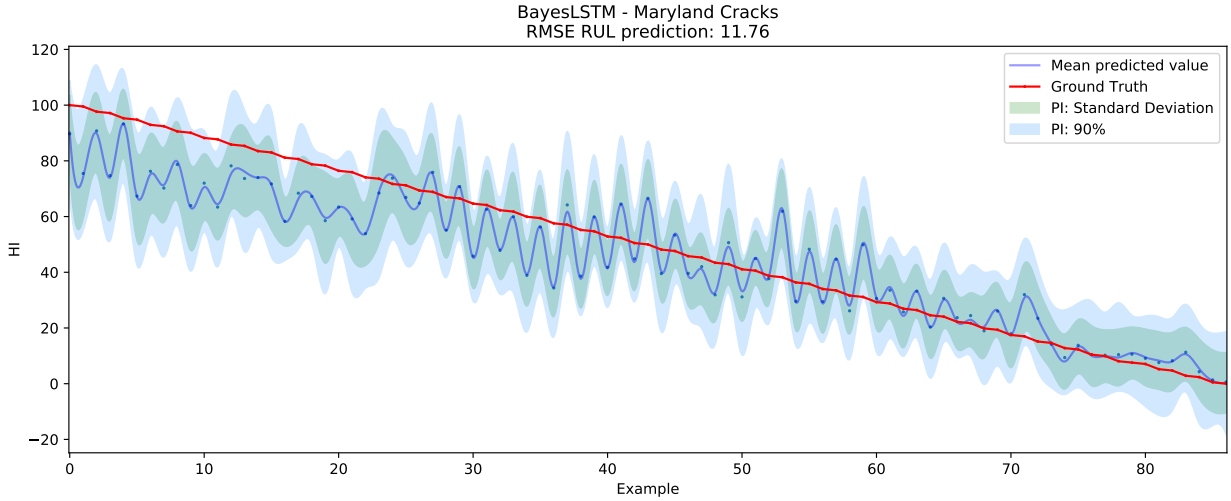


Figure G.7: Fourth complete life cycle. Maryland cracklines, Bayesian LSTM.

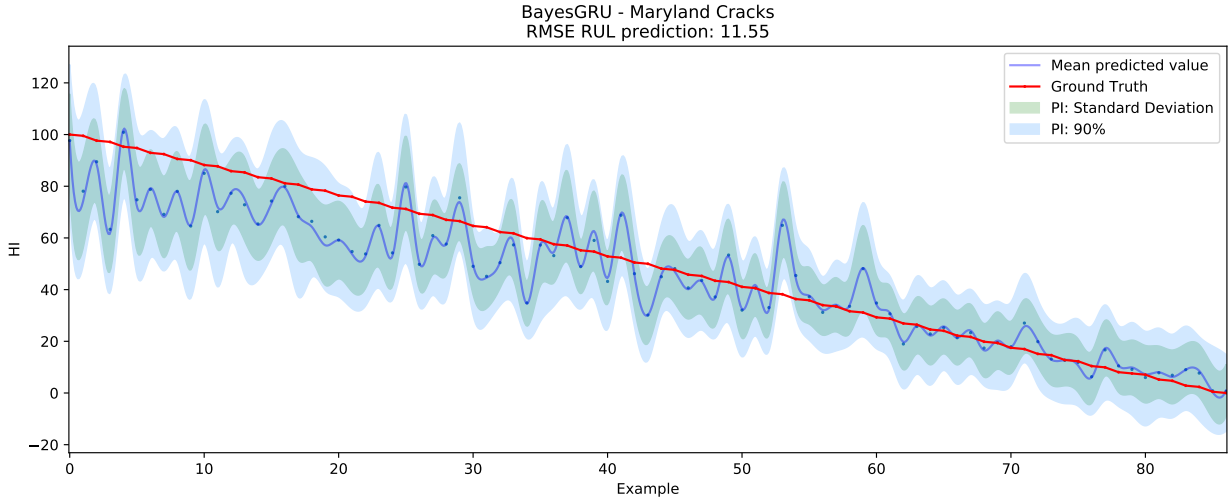


Figure G.8: Fourth complete life cycle. Maryland cracklines, Bayesian GRU.

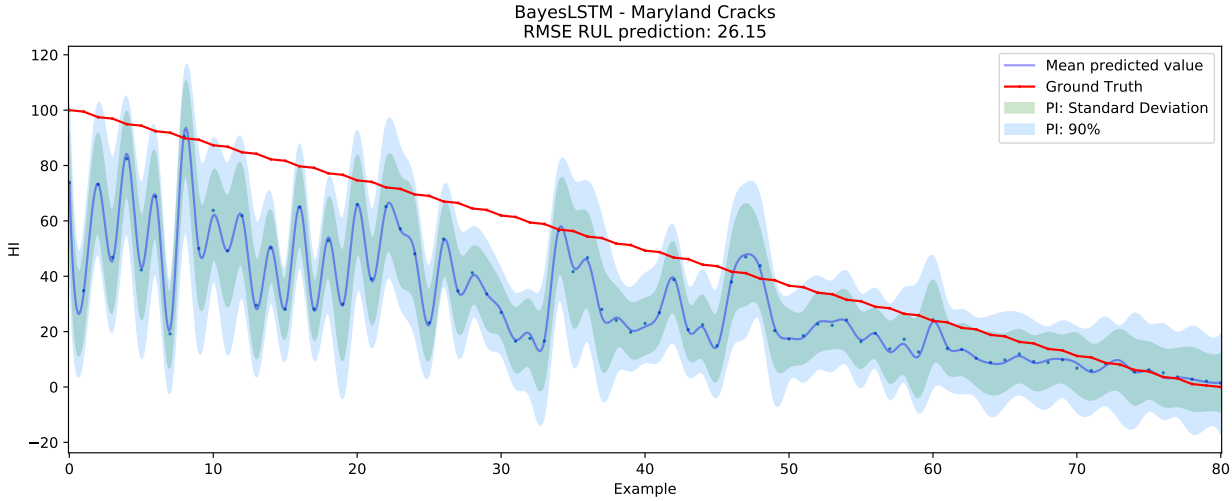


Figure G.9: Fourth complete life cycle. Maryland cracklines, Bayesian LSTM.

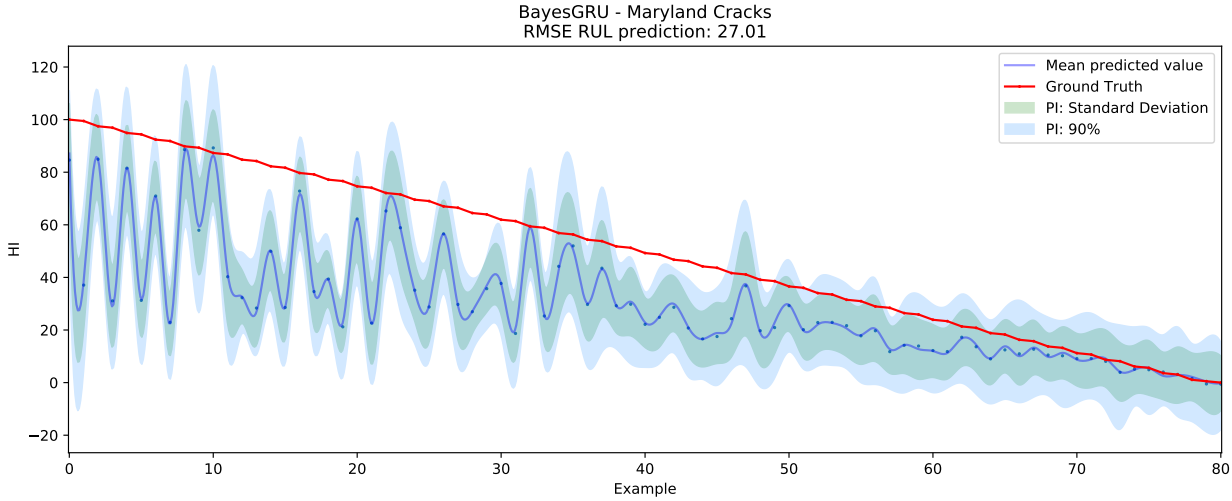


Figure G.10: Fourth complete life cycle. Maryland cracklines, Bayesian GRU.

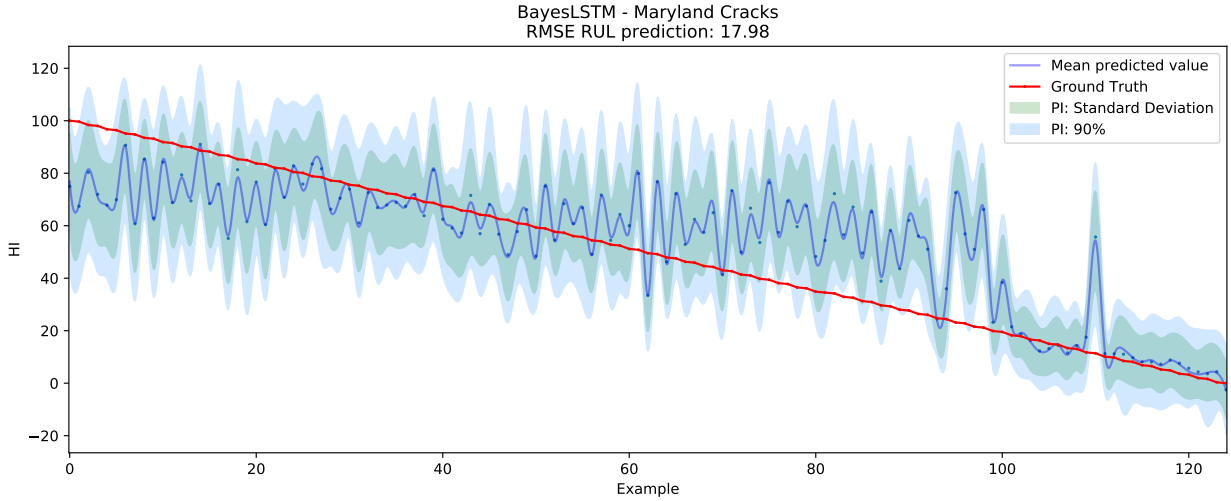


Figure G.11: Fifth complete life cycle. Maryland cracklines, Bayesian LSTM.

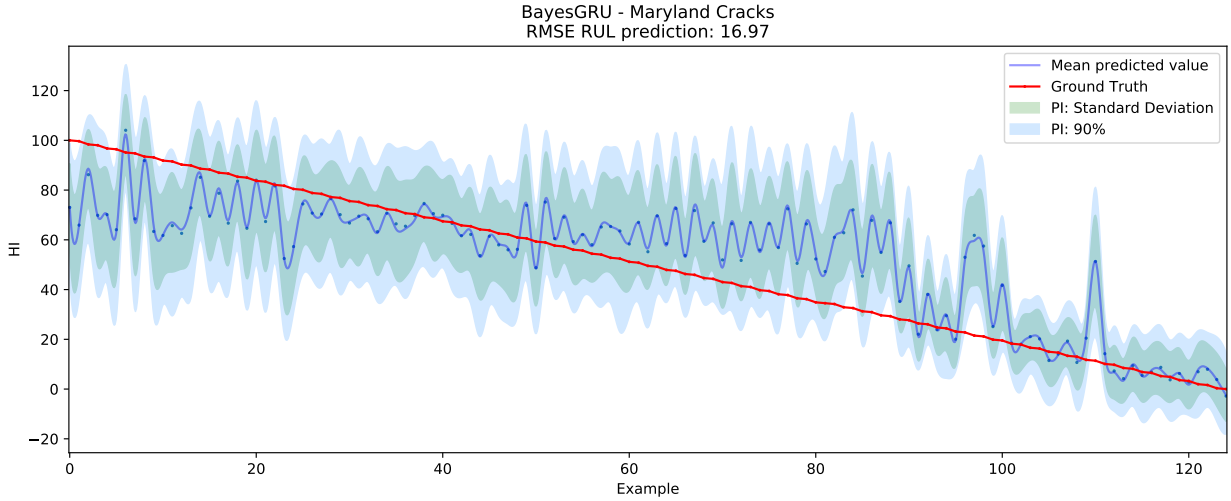


Figure G.12: Fifth complete life cycle. Maryland cracklines, Bayesian GRU.

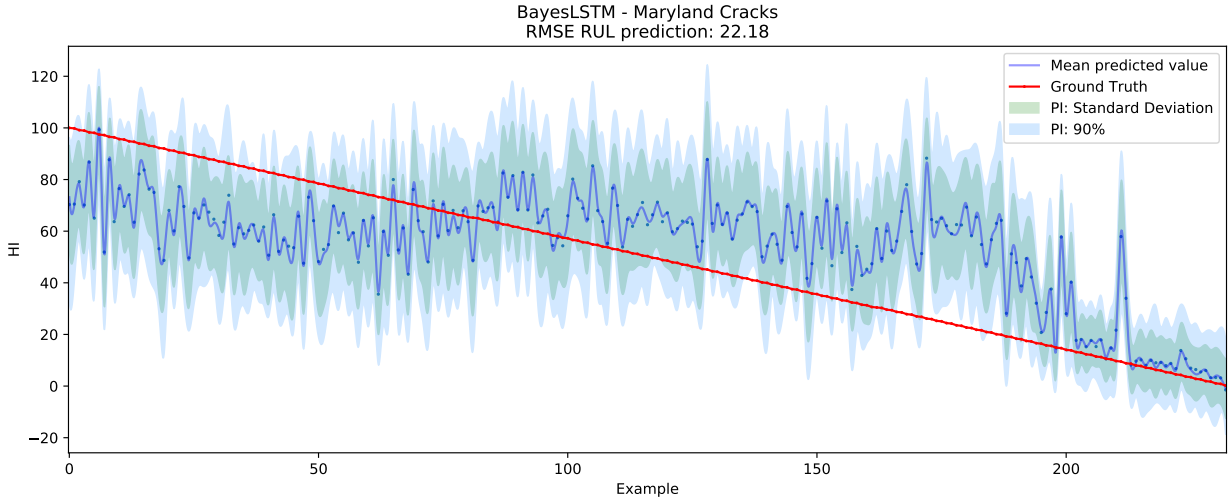


Figure G.13: Sixth complete life cycle. Maryland cracklines, Bayesian LSTM.

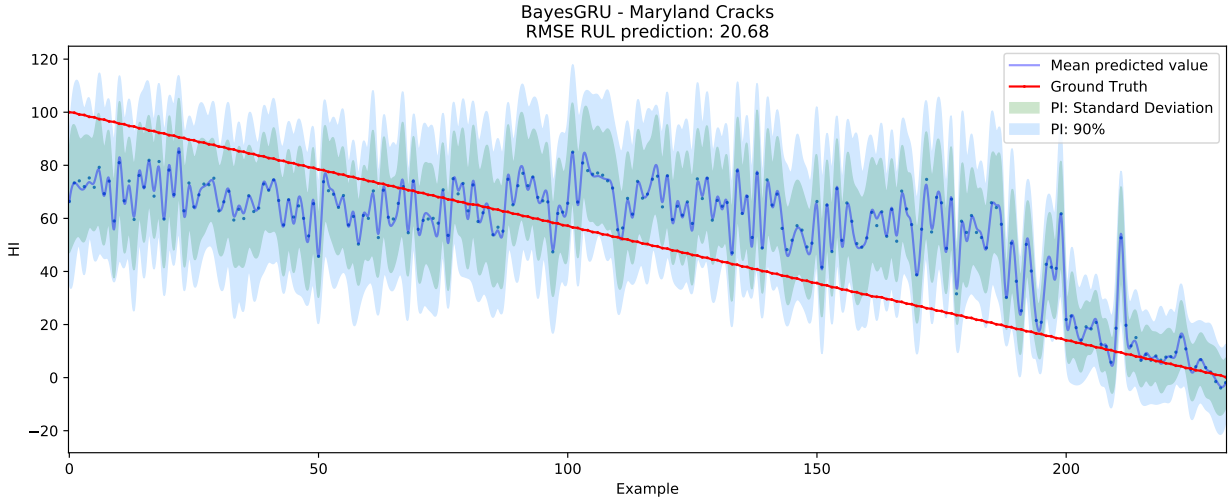


Figure G.14: Sixth complete life cycle. Maryland cracklines, Bayesian GRU.

H Results of Bayesian Recurrent Neural networks, GPM Wind turbine dataset.

Table H.1: Results Wind turbine: Total results.

	Bayes LSTM			Bayes GRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Mean RMSE	13.60	13.30	13.31	13.62	13.88	13.62
STD	0.32	0.32	0.28	0.32	0.30	0.28
RMSE*	8.47	7.98	7.99	8.55	8.92	8.51

I Results of Bayesian Recurrent Neural networks, Ottawa Bearing dataset.

Table I.1: All results for each Bayesian layer. Dataset 1.

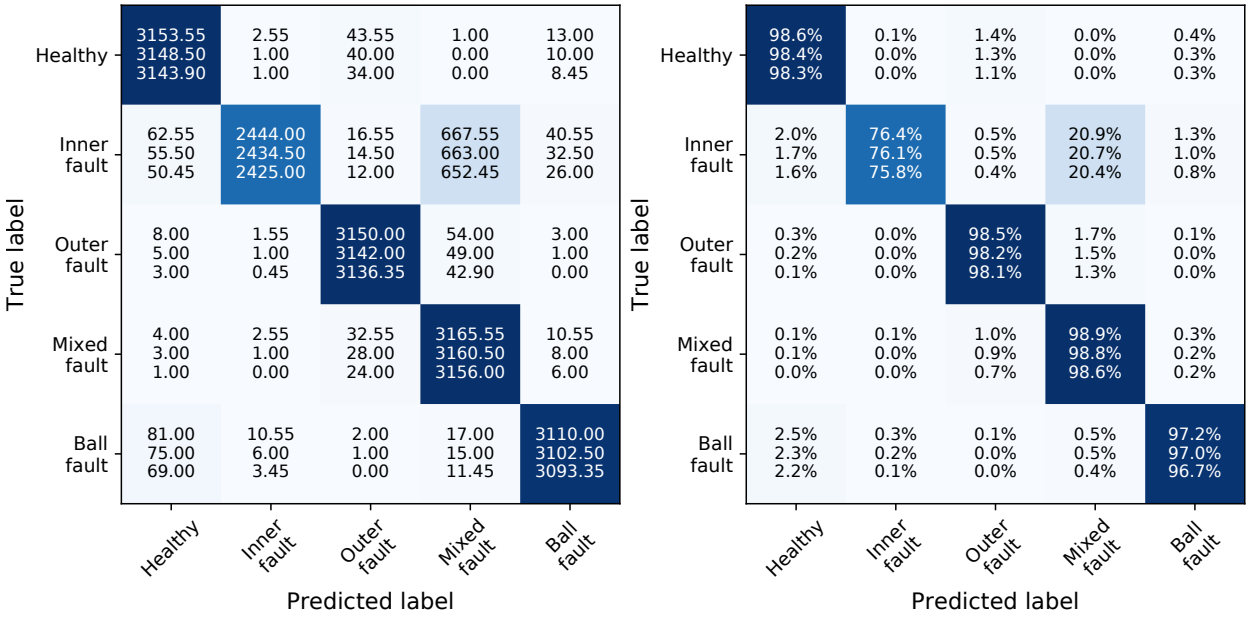
	BayesLSTM			BayesGRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Top 5%	93.73%	93.91%	93.40%	93.85%	94.03%	98.31%
50%	93.55%	93.69%	93.21%	93.67%	93.81%	98.15%
95%	93.32%	93.48%	92.98%	93.44%	93.61%	97.95%

Table I.2: All results for each Bayesian layer. Dataset 2.

	BayesLSTM			BayesGRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Top 5%	98.94%	99.05%	98.94%	99.27%	99.21%	99.19%
50%	98.79%	98.89%	98.80%	99.15%	99.06%	99.05%
95%	98.63%	98.72%	98.66%	99.05%	98.85%	98.89%

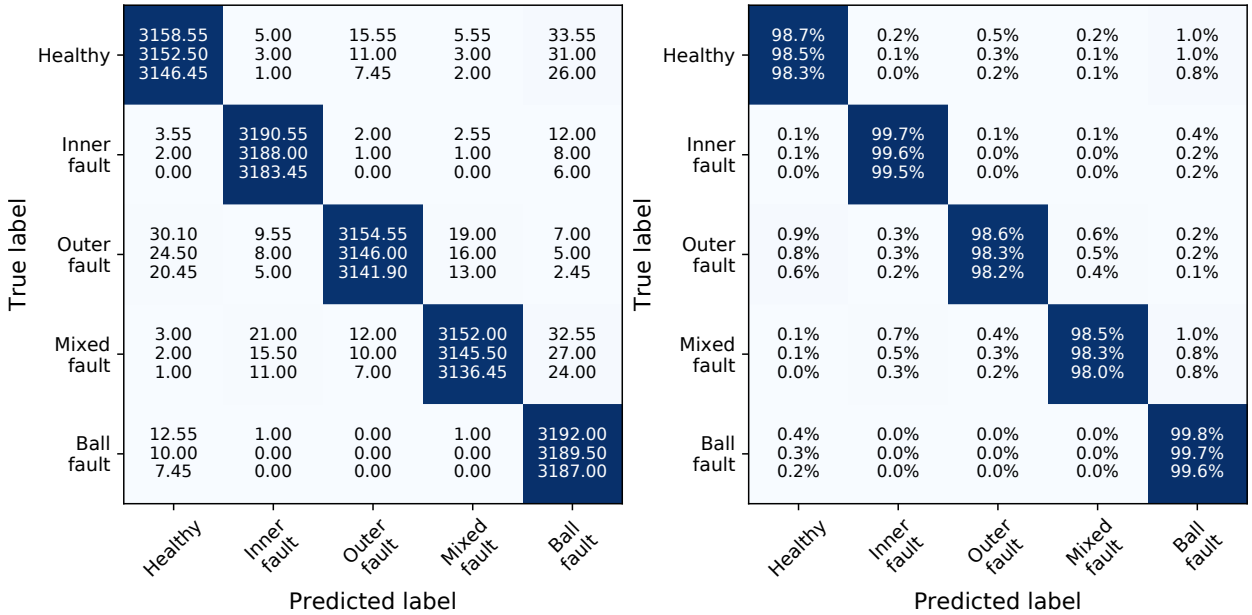
Table I.3: All results for each Bayesian layer. Dataset 3.

	BayesLSTM			BayesGRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Top 5%	97.36%	97.78%	97.39%	97.43%	97.39%	97.82%
50%	97.13%	97.61%	97.25%	97.27%	97.20%	97.65%
95%	96.91%	97.44%	97.05%	97.11%	97.03%	97.45%



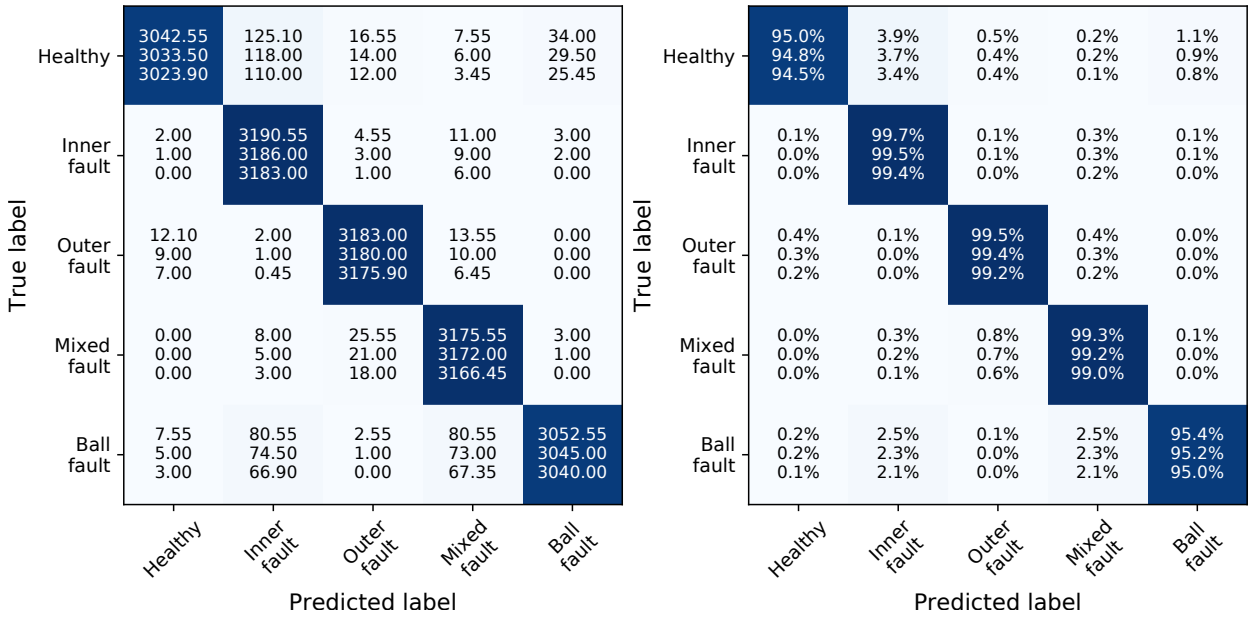
(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure I.1: Best result for Ottawa bearing dataset. Dataset 1 with Bayesian LSTM layers.



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure I.2: Best result for Ottawa bearing dataset. Dataset 2 with Bayesian LSTM layers.



(a) Confusion Matrix (b) Normalized Confusion Matrix

Figure I.3: Best result for Ottawa bearing dataset. Dataset 3 with Bayesian LSTM layers.

J Results of Bayesian Recurrent Neural networks, Politecnico di Torino Bearing dataset.

Table J.1: All results for each Bayesian layer.

	BayesLSTM			BayesGRU		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
Top 5%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
50%	100.00%	100.00%	100.00%	99.92%	100.00%	100.00%
95%	99.50%	99.70%	99.83%	99.53%	99.95%	99.87%