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

SYSTEM-LEVEL PROGNOSTICS AND HEALTH MANAGEMENT: A GRAPH NEURAL NETWORK BASED FRAMEWORK

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

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

ANDRÉS ALFREDO RUIZ-TAGLE PALAZUELOS

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
JOSÉ MIGUEL CARDEMIL IGLESIAS
JUAN TAPIA FARÍAS

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: ANDRÉS ALFREDO RUIZ-TAGLE PALAZUELOS
FECHA: 2019
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

SYSTEM-LEVEL PROGNOSTICS AND HEALTH MANAGEMENT: A GRAPH NEURAL NETWORK BASED FRAMEWORK

Nowadays, crucial industries that deliver essential services for modern society such as power-plants, oil and gas platforms, pipelines and transportation systems among others can be catalogued as complex systems, showing, commonly, a highly heterogeneous behaviour due to the interactions of the different parts that compose them. Due to this, ensuring its safety and reliability have become a challenging task for Reliability Engineers, being traditional Prognostics and Health Management (PHM) statistical-based approaches not enough to tackle the highly multi-dimensional data coming from these systems. Given this, PHM related problems have presently been tackled using deep learning techniques, which have delivered outstanding results in asset and component analysis. On the other hand, no deep learning based PHM framework has been developed for systems due to its complexity, where it is imperative to not only feed a model with operational information, but also with the explicit relationships between its components, being the newly developed area of graph-based geometrical deep learning a promising tool to do this.

The main objective of this thesis is to propose a deep learning based framework for system-level PHM using graph neural networks, enabling the health assessment of a system in three different levels: system-level, component-level and quantification of component relationships, i.e., mutual impact information. Furthermore, the proposed framework will be validated using a real-world industry system and compared to other nowadays commonly used multi-layer perceptron (MLP) and random forest models.

The methodology used in this work consists mainly of four steps. First, a literature review of nowadays PHM deep learning based frameworks will be done in addition to the review of the latest advancements of graph based geometrical deep learning, with emphasis on graph convolutional networks (GCNs). Second, a GCN-based geometrical deep learning framework will be proposed and explained. Third, a chlorine dioxide generation reboiler system in a cellulose plant will be used as a case study for the validation of the framework, showing the preprocess done to it for its use. Lastly, the proposed framework results is discussed and compared with an MLP and random forest models.

The main conclusion of this work is that the proposed system-level PHM framework shows to be a promising tool for the health assessment of systems, outperforming in general other presently used MLP and Random Forest models in terms of accuracy and performance for both the system-level and component-level. Furthermore, regarding the system component relationships, the proposed framework showed to deliver very valuable and interesting insights regarding the analysis of the mutual relationships among its components.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: ANDRÉS ALFREDO RUIZ-TAGLE PALAZUELOS
FECHA: 2019
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

SYSTEM-LEVEL PROGNOSTICS AND HEALTH MANAGEMENT: A GRAPH NEURAL NETWORK BASED FRAMEWORK

Hoy en día, las principales industrias que entregan servicios esenciales a la sociedad moderna (tales como plantas de generación eléctrica, plataformas de gas y petróleo, ductos y sistemas de transporte) pueden catalogarse como sistemas complejos, presentando una gran heterogeneidad en su comportamiento debido a las interacciones de las distintas partes que las componen. Por esto, promover su seguridad y confiabilidad ha sido un gran desafío para los ingenieros, donde las herramientas comúnmente utilizadas en el contexto de *Prognostics and Health Management* (PHM) no son suficientes para analizar de manera correcta la información altamente multidimensional proveniente de estos sistemas. Dado lo anterior, actualmente, cuando se analiza un problema relacionado a PHM, se utilizan técnicas basadas en aprendizaje profundo de máquinas (DL), las que han entregado excelentes resultados para el análisis de activos y componentes. Por otro lado, a la fecha no se ha desarrollado ningún marco de trabajo de PHM a nivel de sistemas usando DL dada su alta complejidad, donde parece imperativo el uso de no solo información operacional del sistema, sino que también explicitar las relaciones entre sus componentes, siendo la recientemente nacida área de aprendizaje profundo de máquinas geométrico basado en grafos (Graph-GDL) una herramienta prometedora para abordar esto.

El principal objetivo de esta tesis es proponer un marco de trabajo basado en DL para PHM a nivel de sistemas usando redes neuronales por grafos, permitiendo la evaluación de la salud de un sistema en tres niveles diferentes: a nivel de sistema, a nivel de sus componentes y la cuantificación de la relación entre componentes, es decir, de su impacto mutuo. Junto con esto, el marco de trabajo propuesto será validado utilizando un sistema industrial real y comparado con otros modelos actualmente utilizados tales como el perceptrón multi-capa (MLP) y Random Forest.

La metodología utilizada en este trabajo consistió principalmente de cinco partes: Primero, se realiza una revisión bibliográfica concerniente a las técnicas de DL utilizadas actualmente en el contexto de PHM, a lo que se le suma una revisión de los últimos avances en Graph-GDL con énfasis en redes convolucionales en grafos (GCN). Segundo, un marco de trabajo basado en GCNs es propuesto y explicado. Tercero, un sistema de reboiler para generación de dióxido de cloro en una planta de celulosa es usada como caso de estudio para validar el marco de trabajo propuesto, mostrando el pre-procesamiento requerido para su uso. Finalmente, los resultados asociados al marco de trabajo propuesto con discutidos y comparados con los modelos de MLP y Random Forest.

La principal conclusión de este trabajo es que el marco de trabajo propuesto para PHM a nivel de sistemas parece ser una herramienta prometedora para la evaluación de la salud en sistemas, superando otros modelos actualmente utilizados en términos de precisión y desempeño tanto a nivel del sistema como al nivel de los componentes. Mas aún, este marco de trabajo entrega información altamente valiosa e interesante para analizar las relaciones mutuas entre los componentes que conforman el sistema estudiado.

Gym ergo sum...

Acknowledgments

En primer lugar quiero agradecer, como siempre, a mi profesor guía Enrique López Droguett, quien siempre estuvo ahí para resolver mis dudas y guiarme en este trabajo. Cómo no agradecer las excelentes y enriquecedoras experiencias que he vivido gracias a usted. Muchas gracias.

Por otro lado quiero agradecer a mi familia. A mis papás por su constante apoyo y cariño; ustedes me dieron la oprtunidad de ser libre, nunca necesité nada ¡y no solo eso! me enseñaron que soy responsable de las decisiones que tomo. Nunca me privaron de nada, siempre conversamos todo y desde chico me hacían reflexionar sobre lo que hacía. No saben cuanto cuánto se los agradezco; los quiero mucho. ¡Y obviamente a Gonzalo! ¿tengo algo que no agradecerte? no solo eres mi hermano, sino que también mi mejor amigo ¡unos años más y compramos el bote!.

A mis amigos de siempre, Valdés, Schaad y Brian; creo que ya tienen claro lo agradecido que estoy de tenerlos como parte esencial de mi vida. Desde hace prácticamente 20 años nos han unido sinfín de historias ¿cómo puedo ser tan afortunado de poder seguir la historia con los mismos personajes principales? muchas gracias.

A mis amigos de la universidad Pipe, Vale y Gabi; hicieron que estos sean de los mejores años de mi vida, por eso les dedico un cariñoso "Aló, toc toc, tontitos...". A los san martines (que asco) Jean, Richi, Sibona, Seba, Danilo, Chipa y Gabriel gracias por la constante alegría; subirme y viajar en la Sasha con ustedes fue un privilegio. También agradezco a los de la salita de magíster por compartir tantas risas conmigo ¿un coupcito?. Y obviamente agradecer a la Manu, lo único que haz hecho es llenar de felicidad mis días.

Finalmente, quiero agradecer especialmente a mi abuelo Julio... me enseñaste a contemplar y no solo a mirar. Gracias a ti tengo los ojos abiertos y curiosos.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays, crucial industries that deliver essential services for modern society such as power-plants, oil and gas platforms, pipelines and transportation systems among others can be catalogued as complex systems, showing, commonly, a highly heterogeneous behaviour due to the interactions of the different parts that compose them [1]. Due to this, ensuring its safety and reliability have become a challenging task for Reliability Engineers, being traditional Prognostics and Health Management (PHM) statistical-based approaches not enough to tackle the highly multi-dimensional data coming from these systems.

In the context of PHM, since the use of modern sensing technologies, industries have gathered high amounts of data, enabling in the recent decade the use of big-data techniques to assess the health state condition, time-to-failure prediction and real-time monitoring of assets and components such as engines [13] and bearings [30]. These techniques are based on data-driven approaches, using just historical data from assets to infer and learn meaningful features from them that can be mapped to different metrics that assesses fault diagnosis and prognosis on them, a characteristic that also make this approach easier to generalize on different scenarios. Due to this, many data-driven approaches based on Machine Learning (ML) have been proposed in the recent years to tackle PHM problems, such as Support Vector Machines [25], Decision Trees (DT) and Principal Component Analysis (PCA) [29]. Lately, Deep Learning models (DL) have dominated the scene due to their capability and versatility to automatically extract features and high-levels of abstraction from highly multi-dimensional data without any specific expert knowledge [15], excelling over other approaches in different tasks such as remaining useful life prediction of aircraft turbofans [21] and fault diagnosis of concrete bridges crack detection [24].

Modern crucial industries behaves as complex-systems, which are defined as a system with many components of different natures interacting with each other. Industries associated systems involves dynamic, interconnected and geographically-dispersed machine and human resources, which results on a highly unpredictable and heterogeneous behaviours, with almost no clear relation to how its individual components or sub-systems are behaving. Given this challenges, to expand PHM capabilities for system-level analysis, it is imperative to not

[1]These parts range from specific components and machines to decision-making human teams.

only use directly operating sensors information, but also consider explicitly the relationships between its components.

For the best of the author knowledge, there are no established DL-based frameworks for system-level PHM, being much more common to find models that focus on the health assessment of single components. For the past two years, a new area in DL has been emerging: Geometric Deep Learning (GDL). The main focus of this new trend is to use DL techniques on non-euclidean data structures such as manifolds and graphs, excelling in different fields such as recommender systems and molecule classification [35]. Due to the outstanding capabilities of Convolutional Neural Networks [20] to extract features from euclidean data structures such as images in computer vision tasks [12], [34], [33], generalizing the convolution operation in CNNs to non-euclidean domains have become a research focus recently, showing promising results in the are as it can be seen in Bronstein et. al [4] review paper on the subject. For the case of industry related systems, its components and relationships can be modelled as a graph structure, opening the possibility of developing a framework where both system component features and their relationships can be processed in a single framework. Furthermore, working with GDL on graphs, i.e., using Graph Neural Networks (GNN), enable to asses the health state of a system in different levels: the whole system as a graph-level analysis, system components as node-level analysis and component relationships as link predictions on graphs, this is, quantifying the mutual influence of impact between two components given their degradation processes.

In this thesis, a graph-based deep learning framework is developed exploring the capabilities of convolutions in non-euclidean domains, specifically using the generalization proposed by Kipf et. al. [16] as Graph Convolutional Network (GCN). This framework is made to work on three different levels for a system health assessment: (1) System-level anomaly behaviour detection, (2) Component-level anomaly behaviour detection and (3) Component relationships for healthy and anomaly behaviour scenarios of the system. To validate the proposed method, data gathered for 12 years from different components of a cellulose plant reboiler sub-system is used.

1.1 Motivation

The present need to extend PHM capabilities to modern-day systems to asses its health-state is still an open problem on the reliability engineering community. Data driven approaches based on DL has shown remarkable results on component-level PHM, so its application on a system-level framework is highly interesting and relevant in today industry context. Nowadays, high amounts of data from different data sources are available for systems health assessment, but due to their highly heterogeneous behaviour, a framework that integrates these data keeping the relationships between its individual components is needed, being GNNs a very promising tool to do this. In particular, since the remarkable performance of CNNs in PHM, this work aims to explore the use of graph convolutions in the form of GCNs on system data represented as a graph data structure, developing a framework that can work on both system and component level health assessment and, furthermore, give us information of component relationships in different health scenarios, being hopefully a relevant approach for future research in systems PHM.

1.2 General Objective

Develop a deep graph convolutional neural network based framework for system-level prognostics and health management using multi-sensor fusion.

1.3 Specific Objectives

- Pre-process data from a system as a graph data structure that can be used by a GNN-based model for system-level health assessment.
- Develop and validate a GCN-based model for system-level anomaly behaviour detection.
- Develop and validate a GCN-based model for component-level anomaly behaviour detection.
- Develop and validate a GCN-based model for component-level relationships when the systems behaves abnormally, this is, quantifying the mutual influence of impact between two components given their degradation processes.

1.4 Scope

This thesis consists in the exploration of GCNs for system-level PHM, developing a framework for health assessment on industry systems data. Furthermore, the proposed model will work on three different levels: system-level, component-level and in the prediction of relationships between components for different scenarios.

Chapter 2

Methodology

In order to successfully explore the capabilities of GDL and graph convolutions for system-level PHM, and therefore fulfill the objectives of this thesis, the following steps are followed.

2.1 Literature review and background

The first step to be followed is to find literature on both DL-based methods applied to Reliability Engineering and GDL advancements in the recent years. On one hand, DL has been successfully tested and validated on component-level PHM, so many ideas can be gathered in order to manage massive amounts of data for health assessment, which is crucial to develop a system-level PHM framework. On the other hand, for the best of the author knowledge, GDL have not been applied yet in Reliability Engineering. Anyway, for the past two years, even though GDL is still a young area of research in DL, big advancements have been made, specially in the generalization of convolutions on non-euclidean data structures such as graphs. Due to this, it is necessary to review the different approaches and methods developed until now in this field, to select the one that best fits to the objectives of this work.

2.2 Geometric Deep Learning framework proposal

Having made a satisfactory literature review, a GDL-based model is selected for the development of a system-level health assessment PHM framework. The proposed framework is made to work on three different levels in the system: First, a system-level anomaly behavior detection model is developed. Second, a component-level anomaly behavior detection model is going to be developed. Finally, a model that predicts component-level correlations for healthy and anomaly behavior scenarios of the system are is going to be developed. For this purpose, Python is used as programming language with its respect scientific computation libraries and machine learning libraries, specifically TensorFlow.

2.3 Case Study: chlorine dioxide generation reboiler system in a cellulose plant

To validate the proposed GDL framework, a real industrial dataset from a chlorine dioxide generation reboiler system in a cellulose plant is going to be used. Composed by five different mechanical assets and seven different operation monitoring sensors, this system is propitious to explore the application of GNNs for its health assessment.

Before everything, an exploration of the dataset is needed due to different factors such as missing data and health state labels to study. Due to this, a ML-based preprocessing is going to be employed using Principal Component Analysis (PCA) and Density-based spatial clustering of applications with noise (DBSCAN). These two techniques will serve the purpose of clustering data in a semi-supervised fashion to identify anomalous behaviors from healthy ones on the studied system.

Finally, to employ a GDL-based model in an engineering system dataset its necessary to model the latter as a graph. To do this, is necessary to define the structure of the graph and its components, such as nodes, links (this will be defined later in this work) and the information that will be contained these two.

2.4 Systems-level PHM framework hyperparameter tuning for training and performance metrics

For each of the three levels of work of the proposed framework, a different model architecture is going to be developed to best assess the problem (defined as modules). Furthermore, an hyperparameter fine-tuning is going to made to fit each model. Finally, to evaluate the performance and robustness of these models, ten separate training runs will be made for each one.

2.5 Results and discussion

To evaluate the proposed framework performance for its different work levels, its important to highlight the nature of the problem that the proposed model is solving, which in this case corresponds to a classification problem with discrete target labels. Due to this, for the three modules that compose the framework, the results will be shown as confusion matrices. Furthermore, to analyze the performance of the framework the Accuracy score is going to be used in addition to Precision-Recall F1, Area Under the Curve (AUC) and Average Precision (AP) scores. Furthermore, the obtained results will be compared to two other types of models: Multi-layer perceptron and Random Forest, ending with a discussion concerning the application of the proposed framework in real industry related systems.

2.6 Resources

To run the proposed framework for systems level PHM the following resources are used.

- Databases from the chlorine dioxide generation reboiler system in a cellulose plant.
- PC with Ubuntu 16.04, i7-7700k processor, 32 Gb RAM and a Nvida Titan X GPU.
- Python 3.7 programming language. Scipy and Numpy as scientific libraries. TensorFlow 1.13, CUDA 10.0, CUDNN 7.5 and Keras 2.2.4 Machine Learning framework.

Chapter 3

Theoretical Background

The following chapter details the necessary background needed to understand and achieve the objectives of this work. Firstly, a quick review is given on the concepts needed to frame this work on the context of ML and DL. Furthermore, some specific techniques used in this thesis will be explained. Secondly, a review on graph theory is made to introduce the reader to the key concepts needed to understand the different GDL-based models that will be reviewed to construct the proposed framework.

3.1 System-Level PHM definition

Systems such as power plants, transportation systems, oil and gas platforms, among others, provide services that are essential for nowadays societies. These systems involve many interactions between its components, such as machines, humans/operators and its associated interfaces. Furthermore, these interactions are geographically-dispersed, highly dynamical, non-linear and in some cases chaotic. Due to this, its a major challenge to model a full system behavior and asses its health state without taking into account the interactions (and the nature of them) between its components [26][36].

To tackle the aforementioned challenge, its imperative to model a system in a way that the interactions between its components are explicitly given. To do this, systems can be represented as networks, which are a collection of discrete objects and the relationships between them, enabling the use of graph theory to study them. Given this, a system-level PHM analysis and/or framework should be one that handles not only the different data sources as a whole (such as sensor measurements), but also the interactions between its components.

3.2 Machine Learning

Machine Learning (ML) can be described as a set of different algorithms and techniques that enables a model to be *trained* to perform a desired task, automatically learning patterns and features from data to map it to a desired output, such as regression estimation values for prognosis tasks or label classification for diagnosis tasks. The main characteristic of these models is that they do not need to be explicitly programmed to solve problems, needing only a *training dataset* to train the model for an specific task and a *testing dataset* "unknown" to the model to validate its performance. Its important to notice that a dataset is composed by independent multi-dimensional (i.e. containing multiple features) data $\mathbf{x}$ and, in some cases (as it will be explained below), a set of dependant targets $\mathbf{y}$.

Commonly, the type of problems solved by ML models can be divided into classification and regression problems. Furthermore, ML models can be trained in three different scenarios: fully-supervised, semi-supervised and unsupervised training, regarding the amount of prior known targets to the data is available, such as health state labels or prognosis metrics.

3.2.1 Regression and Classification

In a *regression* problem, a ML model needs to be trained to solve the following problem: given a training dataset $x = \{x_1, ..., x_n\}$ with $x_\text{i} \in \mathbb{R}^N$, find a mapping function $f(\mathbf{x})$ that can estimate the continuous variable $y = f(x) \in \mathbb{R}$ in the most accurate way.

In the context of reliability engineering and PHM, this type of models are used to solve principally prognosis tasks, such as remaining useful life and mean time to failure estimation, among others.

On the other hand, in a *classification* problem, a ML model need to be trained to solve the following problem: given a training dataset $x = \{x_1, ..., x_n\}$ with $x_\text{i} \in \mathbb{R}^N$, find a mapping function $f(x)$ that can map x_i to a discrete class $C_\text{i} \in \mathbb{N}$ in the most accurate way. It is important to mention that a *binary-classification* problem corresponds to a set of disjoint class labels $C = \{C_1, C_2\}$ and a *multi-class classification* problem corresponds to a set of disjoint class labels $C = \{C_1, C_2, ..., C_k\}$.

In the context of reliability engineering and PHM, this type of models are used to solve principally diagnosis tasks such as fault identification and isolation.

As it can be seen, the problem that the proposed framework of this thesis wants to solve is a binary-classification problem, aiming to diagnose if a system is presenting or not an anomalous behavior, where the latter can be represent as a set of discrete disjoint classes $[0, 1]^T$. This notation for the classes labels is defined as *one-hot representation*.

3.2.2 Supervised, Semi-Supervised and Unsupervised training

When developing a ML model, three possible scenarios regarding the amount of prior known training targets arises. Firstly, a *fully-supervised training* scenario is the most ideal one, where a training dataset contains a prior known target y_i for each training data sample x_i, so, when training the model, its easier to fit the training data to a target function. Examples of supervised ML techniques are Support Vector Machines (SVM) for classification and Random Forests (RF) for regression. Secondly, a *semi-supervised training* scenario is a very common one in real industry datasets, where only some prior known targets are available for different training data samples. This scenario can be solved in two possible ways: first, using the available prior targets, develop a model to obtain the unknown targets, so when a ML model is used, a fully-supervised training can be done. On the other hand, a ML model can be designed to be trained with just some prior known training targets, but, as expected, the training is more difficult compared to a fully-supervised one. An example for semi-supervised ML technique is Transductive Support Vector Machines (TSVM). Lastly, another typical scenario present on real industry datasets is a *unsupervised training* one, where no prior targets are known. A common application of ML in this scenario is unsupervised clustering, where different ML models such as k-Nearest Neighbors (kNN) are used on data to form clusters in it which then can be used to label it (i.e. create a "handcrafted" target function for the data).

3.2.3 Machine Learning for data exploration

When a ML-based model wants to be used on a dataset for a specific task, the dataset need to be explored so then it can be preprocessed in a way that best fits the used ML technique. For example, is known that in general less but more informative data increases the accuracy of ML models (specially in classification tasks) [14][3] , so different techniques (that could also be based on ML) can be used to reduce the dimensionality of a dataset. Another preprocessing example, when on a semi-supervised or unsupervised scenario, is it possible to "handcraft" target functions for a dataset such as labels in a classification problem using ML, obtaining information of different possible clusters in data. Also, it is preferable that a dataset is scaled, so large-valued features (such as temperature sensors measured in $[°C]$) do not suppress the information that low-valued ones (such as vibration signals in $[\mu m]$) can contain when a ML model extracts information from them.

Three preprocessing techniques relevant to this work are going to be explained.

Feature Scaling

As explained above, to avoid the mistake of overfitting a ML model to large-valued features from data leaving aside low-valued ones, a preprocess of scaling, is applied on the different features that compose the data x. Two common scaling techniques are used: *Min-Max Scaling* is used to rescale features into the ranges $[0, 1]$ or $[-1, 1]$ depending on the dataset. The general formula for the rescaled data x' is shown in equation 3.1. Also, *Standard Scaling* is used when the mean $\bar{x}$ and standard deviation $\sigma(x)$ of data wants to be used for rescaling features as shown in equation 3.2.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

$$x' = \frac{x - \bar{x}}{\sigma(x)} \tag{3.2}$$

Principal Component Analysis

Principal Component Analysis (PCA) is a ML technique used mainly for dimensionality reduction, this is, reducing the number of features from a dataset $X \in \mathbb{R}^{N \times D}$ where N and D are the number of samples and features respectively contained in the dataset. To do this, a *Transformation Matrix* $T_K \in \mathbb{R}^{D \times K}$ is used to reduce the number of original features from D to K, generating a *Reduced Data Matrix* $Z \in \mathbb{R}^{N \times K}$. These new K features are linearly uncorrelated and named *Principal Components.* The above mentioned transformation is done as seen in equation 3.3.

$$Z = XT_K \tag{3.3}$$

To produce the optimal reduced representation Z its necessary to find a set of coefficients for T_K, being constructed using equation 3.4,

$$T_K = \begin{bmatrix} \\ v_{\Sigma_{\text{i}}} & \dots & v_{\Sigma_j} \\ \\ \end{bmatrix} \tag{3.4}$$

where the set $\{v_{\Sigma_{\text{i}}}, ..., v_{\Sigma_j}\}$ corresponds to the eigenvectors associated with the set of largest K eigenvalues of the *Covariance Matrix* Σ_X of X. Its important to notice that a subset $l \subset K$ can represent almost all the variance of the original dataset, thus allowing the use of just l principal components for a better representation of the reduced data.

Density-based spatial clustering of applications with noise

Density-based spatial clustering of applications with noise (DBSCAN) is a clustering algorithm that enables the grouping of data samples that are close-packed together in a density regions, leaving samples in low-density regions as outliers (or noise). Two parameters are manually set to find clusters: a distance ε and the minimum number of points inside the reach of this distance S to be considered as a cluster from a central data sample (*core*).

For clusters to be made, its necessary to identify *core points*, *reachable points*, *direct-reachable points* and *noise points* as follows:

- A data sample is a *core point* C if at least S points are in the reach of ε from it (including itself). All points inside ε from C will be considered as a *direct-reachable point* C'.
- A *reachable point* C' is one that is not necessarily a *core point* but a path of *core points* reach to it, this is, i *core points* are *direct-reachable points* between each other.
- A *noise point* N is a data sample that is not *reachable* from any other point C or C'.

Now, a cluster is made by a *core point* and all the *reachable points* that shares paths with it. In Figure 3.1 the latter algorithm is shown.

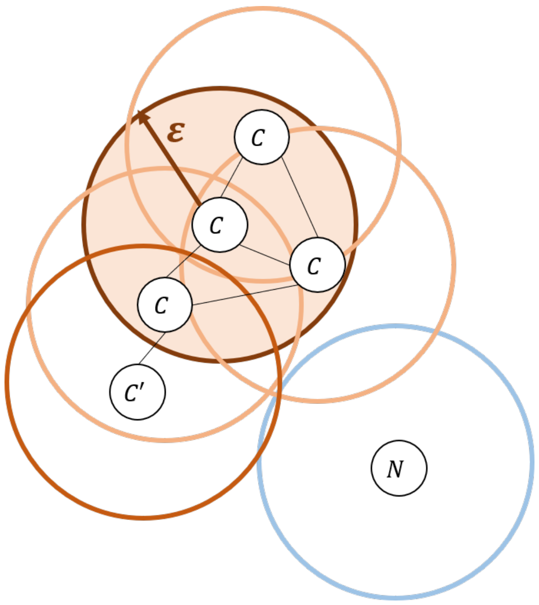


Figure 3.1: DBSCAN clustering algorithm for $S = 2$. The set $\{C, C'\}$ conform a cluster and N represents noise.

In a classification problem with a semi-supervised training scenario, parameters ε and S can be "hand-tuned" to create clusters in a dataset that best fits the prior-known labels, enabling the full dataset to be labeled.

3.2.4 Random Forest models

Random Forests its a widely used machine learning model for both classification and regression tasks. They work by generating multiple decision trees that randomly use different features from the input data and changing the relevance of them, with the aim of optimizing the performance of the task by summarizing the results obtained by each tree as output of the model. For an in-depth look, the reader is encouraged to read the following references [8] [3].

3.3 Deep Learning

Deep Learning (DL) is a subfield of ML that emphasizes in the use of successive "layers" of increasingly meaningful representations from data, using "neural networks" as building-blocks to do this. These techniques are of special interest due to their capability and versatility in automatically extract features from multi-dimensional data and then map it to a target function, such as an asset or system health state or metric in the context of PHM.

Basically, a DL model is commonly composed by the following parts:

- **Input layer:** a chosen data structure to train the model (for example, an image, time-window of operating sensors, among others).
- **Hidden layers:** all the neural network-based operations are performed in a series of stacked hidden layers. Commonly, a deep learning model consists of more than two of them, following the intuition that more layers will extract more meaningful information from data. A series of trainable variables named "synaptic weights" or just "weights'" are present in these layers.
- **Output layer:** the output of the network is obtained with the aim of comparing it to the ground true targets (such as health-state labels) using a loss function.
- **Loss function:** a function that compares the results obtained in the output layer with the ground truth targets. Its value is used as a feedback to train the model.
- **Optimizer:** to train the model, the value obtained by the loss function is used to update the weights of the hidden layers, so a better result is obtained in a following training run of the model.

The previous definitions can be seen in the flowchart shown in Figure 3.2.

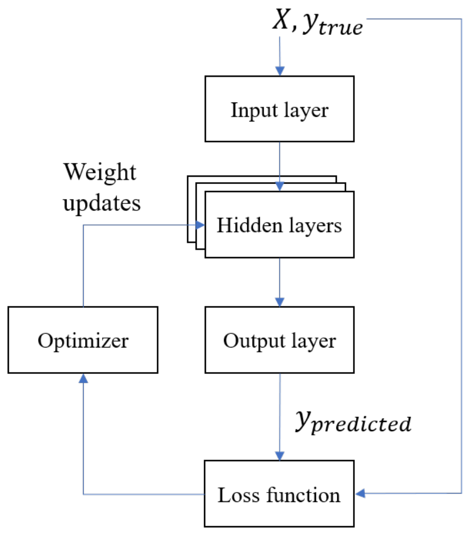
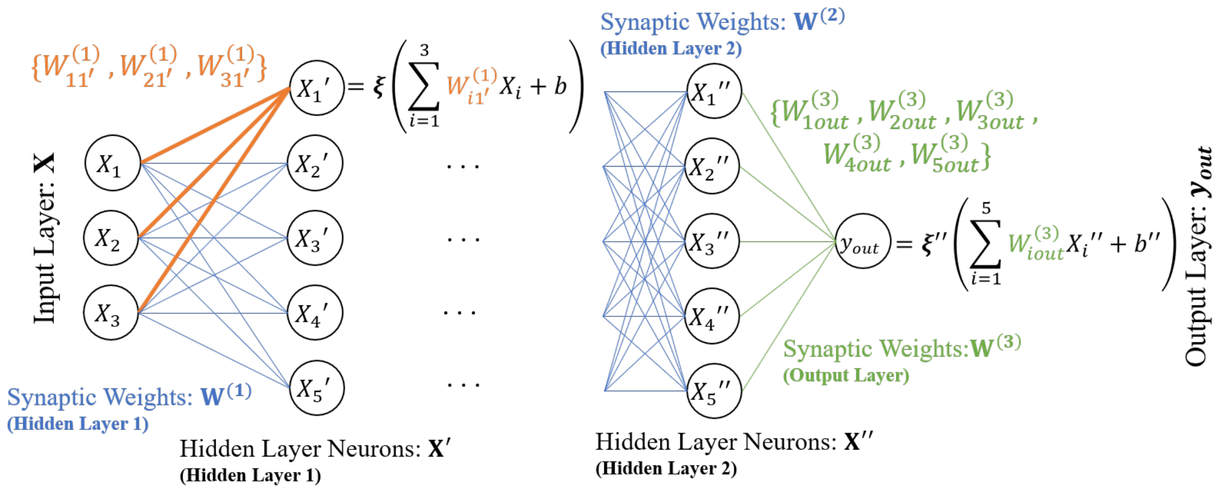


Figure 3.2: Basic structure of a DL model.

3.3.1 Artificial Neural Networks

Artificial Neural Networks (ANN), also known as Multi-Layer Perceptron (MLP), are the building blocks of DL models. They consist of a set of *neurons* in multiple hidden layers that operates the input data of its corresponding layer (that could be preceded by the input layer or a previous hidden layer), ending in the output layer that predicts the target the model training, such as a health-state label for classification problems. A schematic illustration of a 5-layer MLP model (consisting of an input layer, two hidden layers and an output layer) is presented in Figure 3.3.



$\{W_{11'}^{(1)}, W_{21'}^{(1)}, W_{31'}^{(1)}\}$
$X_1' = \xi\left(\sum_{i=1}^{3} W_{i1'}^{(1)} X_i + b\right)$
Synaptic Weights: $\mathbf{W}^{(2)}$
(Hidden Layer 2)
$\{W_{1out}^{(3)}, W_{2out}^{(3)}, W_{3out}^{(3)}, W_{4out}^{(3)}, W_{5out}^{(3)}\}$
$y_{out} = \xi''\left(\sum_{i=1}^{5} W_{iout}^{(3)} X_i'' + b''\right)$
Input Layer: $\mathbf{X}$
Output Layer: $\boldsymbol{y_{out}}$
Synaptic Weights: $\mathbf{W}^{(1)}$
(Hidden Layer 1)
Synaptic Weights: $\mathbf{W}^{(3)}$
(Output Layer)
Hidden Layer Neurons: $\mathbf{X}'$
(Hidden Layer 1)
Hidden Layer Neurons: $\mathbf{X}''$
(Hidden Layer 2)

Figure 3.3: Schematic illustration of a 5-layer MLP model.

As it can be seen from Figure 3.3, each neuron X_{i} performs a linear combination of its input weights coming from a previous layer. Then, the neuron is "activated" by a non-linear *activation function* ξ that fits the extraction on non-linear features from the input data, considering that commonly multi-dimensional data used to train DL models is highly non-linear. More on activation functions is going to be explained later in this section.

An example of a single neuron operation such as X_1' in the hidden layer 1 in Figure 3.3 is shown in equation 3.5,

$$X_1' = \xi\left(\sum_{\mathrm{i}=1}^{3} W_{\mathrm{i}1'}^{(1)} X_{\mathrm{i}} + b\right) \tag{3.5}$$

where $W_{\mathrm{i}1'}^{(1)}$ corresponds to a set of trainable weights coming from each neuron of the previous layer i $\in \{1, 2, 3\}$ and b is a bias term, which consist of a low-valued number (in the order of 0.01), that is used to activate a neuron even though its correspondent weights are 0. Then, using Figure 3.3 as reference, it can be seen in equation 3.6 in matrix notation that an MLP model is just a function composition of operations performed by sets of neurons in stacked layers that, deeper in the model, extract more relevant features from the input data.

$$y_{out} = \xi'' \left(W^{(3)} \xi' \left(W^{(2)} \xi \left(W^{(1)} X + b \right) + b' \right) + b'' \right) \tag{3.6}$$

Activation Functions

As explained above, a non-linear activation functions must be applied in each hidden layer, or the stack of multiple hidden layers will be equivalent of just having just one layer with more neurons, thus not extracting higher-level representations of data deeper in the model. Commonly used activation functions are presented in Table 3.1.

Sigmoid	$\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$
Softmax	$softmax(x_\mathrm{i}) = \frac{\mathrm{e}^{x_\mathrm{i}}}{\sum_j \mathrm{e}^{x_j}}$
ReLU	$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$
SELU	$SELU(x) = \lambda \begin{cases} \alpha\left(\mathrm{e}^{x} - 1\right) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$
tanh	$tanh(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}$

Table 3.1: Activation Functions.

Some activation functions have become widely popular in the DL community, being a common practice to use ReLU and tanh due to their versatility. Now, when training a neural network (as will be explained below), gradients of the activation functions must be calculated, and functions like ReLU tends to vanish their gradients when networks are deep, which will reduce the model training capabilities (this problem is refered as *vanishing gradients problem*). To avoid the latter, different *regularization techniques* have been developed, being nowadays the use of SELU activation function the most common practice for this, mainly due to the fact that its formulated in a way that there is almost non vanishing gradients when the model is trained. For more information, the reader is encouraged to read the following paper by Klambauer et. al. [18].

Its important to mention that certain practices regarding activation functions are followed for the output layer neurons. If the task to solve corresponds to a regression problem, just one neuron without activation is used on the output layer. On the other hand, in a classification problem, the number of neurons in the output layer is equivalent to the number of different target classes **C**, with a Softmax activation function applied to them. This activation ensures that the sum of the different values of the output layer neurons sums up to 1, thus being equivalent to the probability of existence of each class; in other words, the neuron with the highest output value will correspond to the predicted class of the model.

Training of Neural Networks

As shown above in this section, a ANN output is a non-linear function of a set of adaptive weight parameters W, this is, $y_{out} = f(W)$. To be trained, the ANN must minimize a *Loss Function* L (also known as *Cost*) that compares the output of the model y_{out} with the prior known targets y given an input data X, as shown in equation 3.7.

$$L(W) = f(W|X, y, y_{out}) \tag{3.7}$$

To do this, an iterative numerical method called *Backpropagation* is used based on *Gradient Descent* approach. In a simple way, this algorithm searches for the optimal weights to solve the problem presented in equation 3.8.

$$\nabla L(W) = 0 \tag{3.8}$$

To do this, the weights of the network are updated using the *Stochastic Gradient Descent* (SGD) algorithm as shown in equation 3.9, running the model for e ≥ 1 iterations, commonly known as *epochs*, where all the training data is fed to the model.

$$W^{(l+1)} = W^{(l)} - \eta \nabla L\left(W^{(l)}\right) \tag{3.9}$$

In equation 3.9, l corresponds to a layer and $\eta > 0$ is a hand-tuned variable called *learning rate* that indicates the rate in which weights are updated. Nowadays, new and better optimization algorithms based on SGD are commonly used to train neural networks, such as *Adam* and *RMSprop*, which can be seen in [9].

3.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are an ANN-based model that uses a receptive field named *kernel* K to extract hierarchical patterns from input data though multiple stacked *convolutional layers*. To do this the *convolution operation* ($\star$) is used within a matrix of weights W_k (that represents the kernel K) and the input data $X \in \mathbb{R}^{I \times J}$ from an arbitrary layer as shown in equation 3.10,

$$y_{(\mathrm{i},j)} = (W_k \star X) = \sum_{a=0}^{w} \sum_{b=0}^{h} W_{k(a,b)} \cdot X_{(\mathrm{i}+a, j+b)} \tag{3.10}$$

where $y_{(\mathrm{i},j)}$ corresponds to the output of the convolution of the kernel k, also named *feature map.* w and h are the width and height of k respectively. i, j corresponds to an specific point $(\mathrm{i}, j) \in I \times J$ from X. As it can be seen, multiple kernels $\mathbf{K} = \{k_1, ..., k_n\}$ can be used on the input data X, so multiple feature maps can be obtained, each one representing a different representation from X.

Given equation 3.10, a convolutional layer output for the kernel k can be expressed as shown in equation 3.11. Furthermore, CNNs kernel weights parameters are trained in the same way as MLPs.

$$Y^{(k)} = \xi \left(W_k \star X + b \right) \tag{3.11}$$

An schematic view of a CNN architecture is presented in figure 3.4. As it can be seen, the different feature maps of the last convolution are flattened and then connected to an MLP that perform the desired classification or regression task.

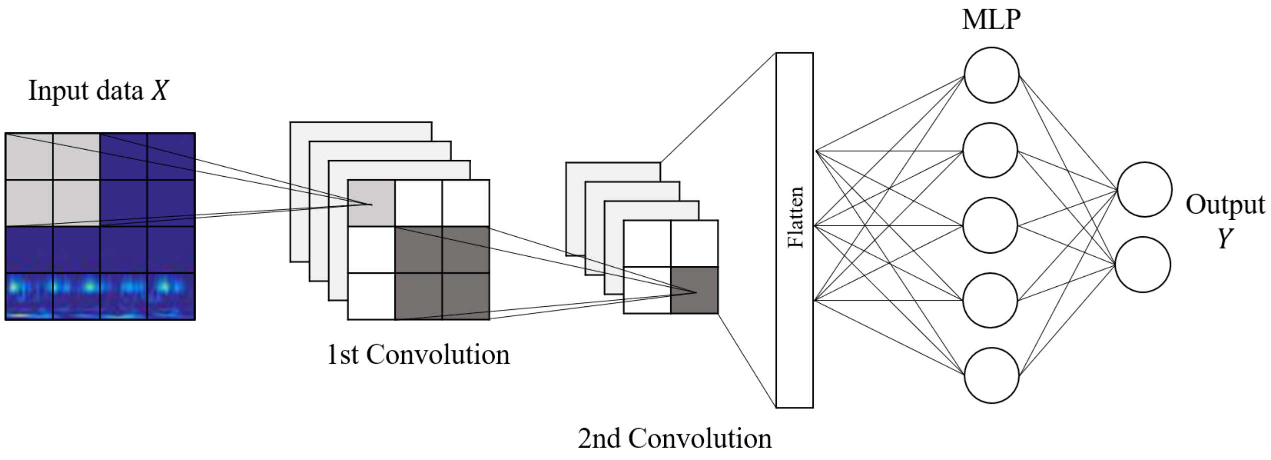


Figure 3.4: Schematic illustration of a two layer CNN architecture.

CNNs outstands in pattern recognition and automatic feature extraction, being widely researched in the reliability engineering and PHM community and excelling in a wide range of problems [15]. The reason for this is that convolutions have the following benefits with respect to MLPs:

- The existence of a constant number of trainable parameters that are trained in each kernel, independent of the dimensions of X.
- Output features from convolutional layers tends to enjoy from translation invariance/-covariance, this is, the convolution kernels from the CNN are insensible for local changes in the input X, without loosing model performance.
- Translational invariance/covariance increases when the network is deeper.

Due to these intrinsic characteristics from the convolution operation and the structure of a CNN [4], many researches are looking to generalize convolutions for other data structures such as graphs and manifolds, a topic concerning the research area of Geometric Deep Learning (GDL) and that, the particular case of graph structures analysis, will be discussed from now on in this chapter.

3.4 Graph theory

Until a few years, DL models where only used on data that belong to euclidean domains, such as images, sensor readings from an asset, among others. Although this approach is suitable to a large range of problems in PHM and reliability engineering, it has a major drawback: the data given to the models to train do not explicitly embed any possible intrinsic structure of the system from where it come from, which is very inconvenient considering that the relationships between its components determine its behavior.

To tackle this problem, an extensive research have been made to apply DL on non-euclidean domains such as graphs and manifolds, which are data structures that explicitly maintain the intrinsic relationships present on systems data such social networks, transportation and energy systems, industrial production plants, among others [28].

Due to the latter, representing a system as a graph seems convenient to analyze its behavior and health, so to understand the recent applications of DL to graph data structures, some graph theory needs to be explained.

3.4.1 Definitions

A *graph* is defined as a pair (ν, ε) where $\nu = \{1, ..., N\}$ is a set of N *vertices* (or *nodes*) and $\varepsilon \subseteq \nu \times \nu$ is a set of *edges* (or *links*). A graph is called *undirected graph* when $(\mathrm{i}, j) \in \varepsilon$ iff $(j, \mathrm{i}) \in \varepsilon$. To describe a graph, the following matrices are defined:

Adjacency Matrix (A)

Its a symmetrical matrix that indicate if a pair of nodes are adjacent or not, this is, connected by a link. For $A \in \mathbb{R}^{N \times N}$, its elements are defined by equation 3.12.

$$a_{\mathrm{i},j} = \begin{cases} 1 & \text{if nodes } (\mathrm{i}, j) \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

Degree Matrix (D)

Its a diagonal matrix that represents the number of links present in each node. Then, for $D \in \mathbb{R}^{N \times N}$, its elements are defined by equation 3.13.

$$\mathrm{d}_{\mathrm{i,i}} = \sum_{j} a_{\mathrm{i},j} \tag{3.13}$$

Graph Laplacian (Δ)

Its a matrix representation of a graph that represents the euclidean Laplace operation on graphs. On a function in the node space $f : \nu \to \mathbb{R}$, the laplacian is defined by equation 3.14.

$$\Delta f = (D - A)\, f \tag{3.14}$$

As a geometrical interpretation, Δ can be seen as the average of a function around a node and the value of the function in the node itself.

The graph laplacian can be normalized by equation 3.15. From now on, the *normalized laplacian* will be represented as L.

$$L = D^{-\frac{1}{2}} \Delta D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{3.15}$$

3.4.2 Spectral Analysis on Graphs

The normalized laplacian L of a graph admits an eigendecomposition with a discrete set of eigenvectors U and eigenvalues $\{\lambda_1, ..., \lambda_k\} \in \lambda$, then, L can be expressed as 3.16,

$$L = U \Lambda U^T \tag{3.16}$$

where $\Lambda = \mathrm{di}ag(\lambda_{\mathrm{i}})$.

In the *Spectral Domain*, the eigenvalues $\{u_1, ..., u_k\} \in U$ can be interpreted as Fourier basis functions and λ as the spectrum of L, so a Fourier Analysis can be developed on graphs. In a graph, the *Fourier Transform* of a function in the node space $f : \nu \to \mathbb{R}$ is defined by equation 3.17.

$$\hat{f}(\lambda) = \langle f, U \rangle = U^T f \tag{3.17}$$

Spectral Convolution

As mentioned in section 3.2.2, due to the success of the use of convolutions on DL applied on euclidean domain datasets, it has been of great interest for researchers to generalize convolutions to non-euclidean domain data structures, specially at graphs [32][4].

To generalize the convolution to graph data structures (i.e. non-euclidean domains), first lets explain the *Convolution Theorem*, which states that a convolution ($f \star g$) between any two functions f, g in their spectral domain can be defined as as the element-wise product of their Fourier transforms. Using the laplacian eigenvectors U as Fourier basis functions, a general form of a convolution is shown in equation 3.18 for a signal $x \in \mathbb{R}^N$ (a scalar for every node of a graph) with a filter $g_\theta = diag(\theta = U^T g)$ parameterized by $\theta \in \mathbb{R}^N$ in the spectral domain. This operation is defined as *Spectral Convolution.*

$$g_\theta \star x = U g_\theta U^T x \tag{3.18}$$

In equation 3.18 it can be seen that $U^T x$ is the graph Fourier transform of x. Also, due to the convolution theorem, $g_\theta = g_\theta(\Lambda)$. With the Spectral Convolution defined, it is possible to show some recently proposed applications in DL models, commonly known as *Graph Convolutional Neural Networks.*

3.5 Geometric Deep Learning in Graphs

To apply DL to graph structures different network architectures have been proposed in the past years, being spectral-based graph convolution networks one of the most studied ones [35], where all uses the definition 3.18 for their construction, differing only on the choice of the filter g_θ.

3.5.1 Graph Convolutional Neural Networks

Spectral Graph Neural Network

The first spectral convolution neural network was proposed by Bruna et. al. [5] and named *Spectral Graph Neural Network* (Spectral-GNN), taking a direct application of a traditional convolutional layer (see equations 3.10 and 3.11) and setting $g_\theta = \Theta \in \mathbb{R}^{k \times k}$ as a diagonal matrix set with learnable parameters. Then, Bruna's graph convolutional layer is defined in 3.19,

$$g_l = \xi \left(\sum_{l'=1}^{q} U_k \Theta_{l,l'} U_k^T x_{l'} \right) \tag{3.19}$$

where $x = (x_1, ..., x_p) \in \mathbb{R}^{N \times p}$ is the $p - dimensional$ input and $g = (g_1, ..., g_q) \in \mathbb{R}^{N \times q}$ is the $q - dimensional$ output of the layer (both on the nodes on the graph) and k represents the first k eigenvectors of U.

Evaluating 3.19 has mayor drawbacks, being computationally expensive as multiplication with U is $O(N^2)$ and computing the k-th order eigendecomposition of L might be expensive for large graphs (i.e. obtaining U in the first place). Due to this, other alternatives have been explored to define g_θ, being Chebyshev polynomials approximations a very successful method that will be presented below.

Chebyshev Network

To address the problem of highly expensive computing power concerning Spectral-GNNs, Defferrard et. al [7] proposed the Chebyshev Network (ChebNet) with the use of a K-th order Chebyshev polynomial $T_k(x)$ approximation for the spectral representation of a function, in this case the filter g_θ, as shown in equation 3.20,

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k \left(\frac{2}{\lambda_{max}} \Lambda - I \right) \tag{3.20}$$

where $\theta \in \mathbb{R}^K$ is now a vector of Chebyshev coefficients (which will be the filter trainable parameters of the network) and λ_{max} the largest eigenvalue of L. Chebyshev polynomials are recursively defined as shown in equation 3.21.

$$\begin{aligned} T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x) \\ T_0(x) &= 1 \\ T_1(x) &= x \end{aligned} \tag{3.21}$$

Given equation 3.20, the spectral convolution 3.18 can be defined as

$$\begin{aligned} g_\theta \star x &\approx U \left(\sum_{k=0}^{K} \theta_k T_k \left(\frac{2}{\lambda_{max}} \Lambda - I \right) \right) U^T x \\ &= \sum_{k=0}^{K} \theta_k T_k \left(\frac{2}{\lambda_{max}} L - I \right) x \end{aligned} \tag{3.22}$$

Equation 3.22 is K-localized since it it a K-th order polynomial of the laplacian, this is, it only depends on nodes that are K steps away from the central node. Furthermore, note that this corresponds to a complexity of just $O(|\varepsilon|)$.

1st Order ChebNet or Graph Convolutional Network

With the aim of simplifying 3.22 to be used in deep networks, Kipf et. al. [16] proposed the use of $K = 1$ in a ChebNet, so a convolutional layer will depend linearly on L, just as a typical MLP formulation (see equation 3.5). Furthermore, they approximate $\lambda_{max} = 2$, so equation 3.22 simplifies to $g_\theta \star x \approx \theta_0 x + \theta_1 (L - I)$ and, to reduce the number of trainable parameters to a single weight matrix they assume $w = \theta_0 = -\theta_1$. Then, the convolution operation in 3.22 can be re-written as

$$g_\theta \star x \approx w \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x \tag{3.23}$$

From equation 3.23 it can be seen that $I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ will have eigenvalues in the range $[0, 2]$, which can lead to exploding/vanishing gradients and other numerical instabilities in deep networks. To tackle this, a *re-normalization trick* is going to be used, with $\hat{A} = I + A$ and $\hat{D} = D(\hat{A})$ so its eigenvalues are in the range $[0, 1]$. Now, 3.23 can be re-written as

$$g_\theta \star x \approx w \left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} \right) x \tag{3.24}$$

Now, 3.24 can be used in a convolutional layer for an input data *Feature Matrix* $X \in \mathbb{R}^{N \times D}$ with D : *number of input features* and a model node-level output $Z \in \mathbb{R}^{N \times F}$ with F : *number of output features* (which can be interpreted as the number of "filters" or "feature maps" in a traditional CNN architecture) as

$$H^{(l+1)} = \xi \left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right) \tag{3.25}$$

with,

$$H^{(l=0)} = X;\ H^{(l=L)} = Z \tag{3.26}$$

where $l = \{0, ..., L\}$ are the layers of the network, $H^{(l)}$ are the outputs of each layer, $W^{(l)} \in \mathbb{R}^{P \times Q}$ is a matrix of filter weights with Q feature maps from layer l and P for $l + 1$ and ξ is a non-linear activation function. This formulation is named *Graph Convolutional Network (GCN)* by its authors.

With this re-formulation of a graph convolutional layer, two main benefits are achieved:

- Alleviate overfitting for nodes with very wide node degrees distributions using just a first order localized filter.
- Higher order neighborhoods representations can be achieved by stacking more layers. For example, for a K-th order neighborhood representation, K graph convolutional network layers can be stacked.
- Multiple layers can be stacked to create a deep network, which in general deliver better results [9].
- Computation complexity is of order $O(|\varepsilon|PQ)$ for each layer.

An schematic view of a GCN is presented in figure 3.5.

Due to the benefits of the GCN architecture and its generalization of the convolution operation in graphs, this model is going to be used to develop the framework proposed in this work. GCNs can be used in three different ways that are propitious to this work:

- First, a GCN model can be used as a graph feature extraction module to feed an MLP model for graph classification.

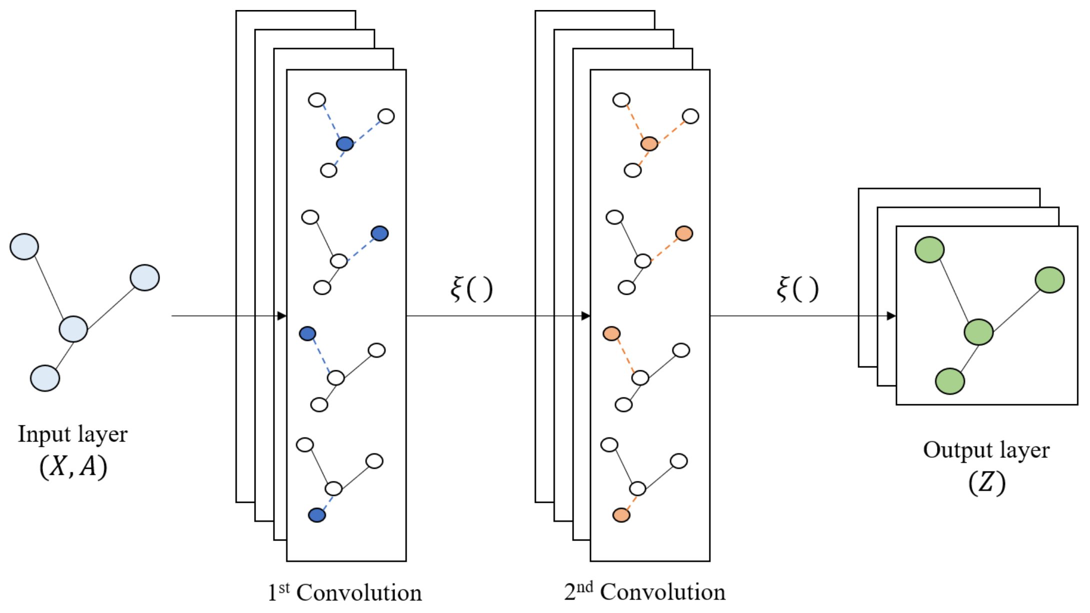


Figure 3.5: Schematic illustration of a two layer GCN architecture.

- Second, a GCN model output features can represent different classes for each node on the graph, enabling node classification.
- Third, a GCN can be used as an auto-encoder [9] to encode a graph in a latent space Z that can be used to re-construct the Adjacency Matrix A of an arbitrary input data X from a graph, enabling link prediction between the nodes that compose it. This approach will be explained with more detail in the next chapter of this thesis.

3.5.2 Review of DL techniques applied in health diagnostics

The proposed framework for system-level PHM will be developed to tackle classification problems, which in the case of PHM it corresponds to anomalous behavior detection and/or diagnosis. To the present day, component-level PHM have been widely studied, obtaining remarkable results in this tasks, specially with the use of CNN-based models. Indeed, Chen et al. [6] used a shallow CNN architecture for gearbox vibration data, obtaining a better classification performance compared to SVMs. Wang et al. [31] used wavelet scalogram images as an input to a deep CNN (DCNN) consisting of convolutional and subsampling layers (i.e., pooling layers for feature dimensionality reduction) for rolling bearing fault diagnosis. Guo et al. [10] implemented an adaptive DCNN for the Case Western's bearing data set [23] to perform fault diagnosis. Verstraete et al. [30] proposed a DCNN architecture for fault identification and classification on rolling bearings in the context of the MFPT [2] and Case Wastern's datasets with minimal data preprocessing, outperforming other deep learning based methods and showing robustness against experimental noise. Modarres et al. [24] proposed a DCNN for detection and identification of structural damage, obtaining strong results on honeycombs structures and concrete bridge crack identification.

For the case of systems, there is a lack of research in building a framework that asseses a whole system health state taking into account the relationships between its components. Furthermore, it is imperative to consider the highly multi-dimensional environment in which a system delivers its operational information. Due to this, DL shows a great potential in tackling this problem [19], but not much applications of DL techniques have been found in literature addressing system-level PHM. The only one that have comparable objectives with this thesis (but do not form part of a system-level PHM framework by the definition presented in chapter 3) is Han et. al. [11] ST-CNN framwork. He proposes an adaptive spatiotemporal feature learning approach for fault diagnosis in complex systems, combining an spatiotemporal pattern network (STPN) approach with convolutional neural networks (CNN) classifier. This framework was applied on a wind-turbine dataset, using a Markov-based model to compute state-transition matrices that include information from state-transition probabilities between sub-systems and temporal measurements, which then are fed to a CNN classifier to identify unseen operational conditions in the system. Even though they show good results in this task, this framework is limited just to a system-level assessment, with no possible insights to other levels in it, such as sub-system or component level health assessment. Furthermore, it could not be considered as a system-level PHM framework by the proposed definition in this work, since the relationships between the components of the studied system are not explicitly given to the model, leaving aside the possible degradation dependencies among them, a highly relevant subject as shown by Assaf et. al. [1]. Given this, for the best of the author knowledge, the framework proposed in this work is the first system-level PHM applying DL techniques in literature.

Chapter 4

Proposed Framework for System-Level PHM

The following chapter details the proposed DL-based systems PHM framework for the following three levels of health assessment: (1) system-level, (2) component-level and (3) determination on relationships of the system components in anomalous scenarios.

4.1 Proposed framework

As stated in the previous chapters, the proposed framework for engineering systems PHM will address three levels of health assessment using three different modules:

1. **System-level module (SLM):** this module will automatically detect if the whole studied system is presenting or not an anomalous behavior that could lead to a system failure.
2. **Component-level module (CLM):** this module will be able to automatically detect which of the components of the system is presenting an anomalous behavior.
3. **System components relationships module (SCRM):** this module will create a *weighted relationships matrix* (WRM) given the information of the system with which the framework was fed. WRM will show how the different components of the system are related to each other when it behaves anomalously.

The main objective of this framework is to give relevant actionable-information for decision makers to assess the health state of an engineering system. To do this, the flowchart presented in Figure 4.1 is proposed as a framework for systems health assessment.

All of the modules that compose the framework are based on GCNs due to the benefits of working with system data represented as a graph, specifically an undirected one (see Chapter 3). This benefits are:

- The relationships (physical or conceptual) between the components of a system can be explicitly fed to a GCN model using the adjacency matrix A. Furthermore, information of the system that is shared between some of its components, such as sensor measurements in piping lines between two assets, can be also be fed explicitly to a GCN model using the feature matrix X.

- A same GCN-based model can perform health assessment in the different hierarchical levels of an engineering system.
- GCNs are built to handle high-dimensional data delivered by nowadays engineering systems.

the system must be represented as a graph with the following characteristics: A, X. To train the model, y is also necessary.

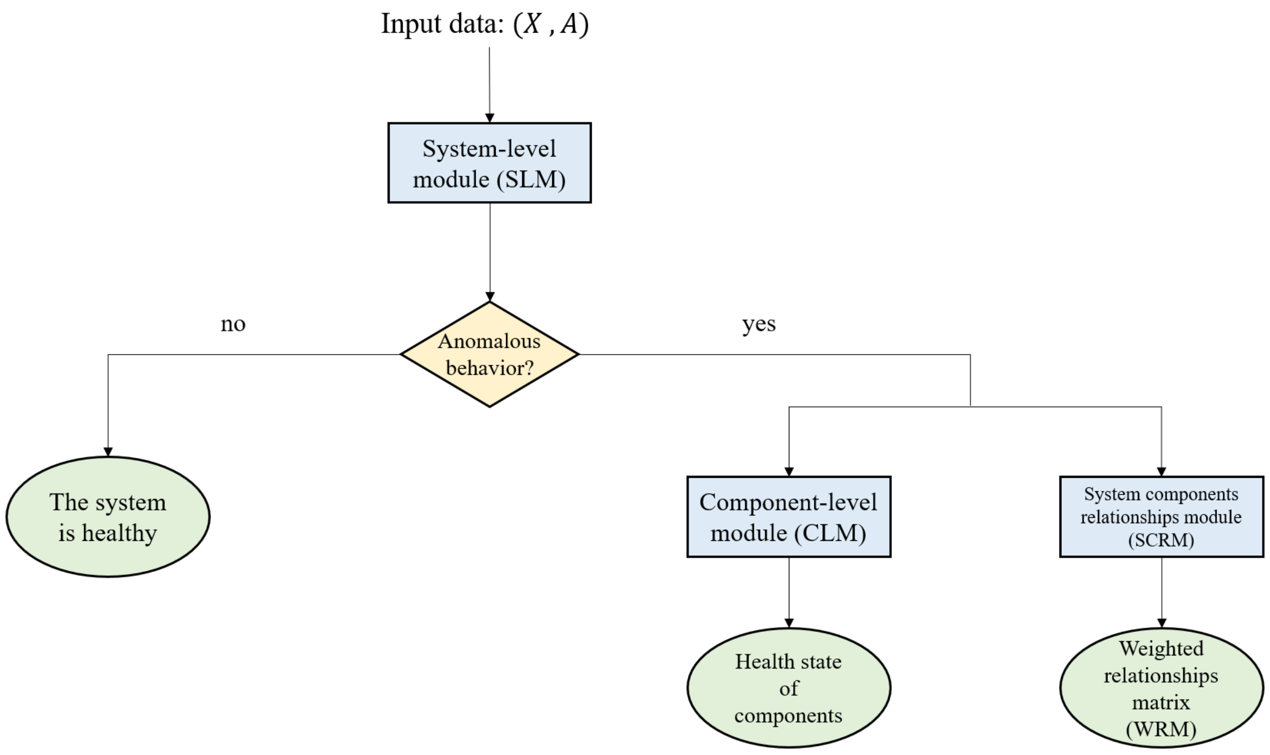

Input data: (X, A)
System-level module (SLM)
Anomalous behavior?
no
yes
The system is healthy
Component-level module (CLM)
System components relationships module (SCRM)
Health state of components
Weighted relationships matrix (WRM)

Figure 4.1: Proposed framework for systems health assessment.

4.2 System-level module

For the **system-level module**, the following steps are going to be performed in a DL architecture to classify if the system behavior is normal or anomalous:

1. A GCN model will be used to extract characteristics directly from the graph information and structure. In this step, multiple features will be obtained for each node of the graph that will contain hidden relevant information of it that is related to the full-system health.

2. The features obtained in the previous steps are embedded in a single vector using a *Self-Attention Pooling* (SAP) mechanism [22], this is, a dimensionality reduction of the GCN output features in a low-dimension embedding containing a summarized representation of it. The implementation of this step will be detailed further in this section.
3. The embedded vector of the GCN features is fed to a traditional MLP model to perform a binary classification task, which in this case will correspond to predict if the studied system is behaving normally or not.

The SAP mechanism introduced previously performs the following operations to generate an embedding of the GCN output features from a graph:

1. The output features of the GCN model $Z = GCN(X, A)$ will have dimensions $n-by-u$, where n is the number of nodes and u the number of extracted features. This is the information that will be embedded using SAP.
2. An *Annotation Matrix* Ω is used to embed Z performing the following operation $\Omega = softmax(W_{s2}tanh(W_{s1}Z^T))$, where W_{s1} is a weight matrix with shape $t - by - u$ and W_{s2} is a dimensionality reduction weight matrix of size $r-by-t$. Here t, is an arbitrary hyperparameter that can be set arbitrarily and r is the number of features that will embed the information of the n nodes in Z. Note that $r \leq n$.
3. Using Ω from the previous step, the embedding vector M of size ru is computed by $M = flatten(\Omega Z)$.

Using the aforementioned steps for the system-level module for systems health assessment, the output of this module will correspond to a binary diagnosis of the system behavior, identifying normal from anomalous. The correspondent DL architecture of this module is presented in figure 4.2.

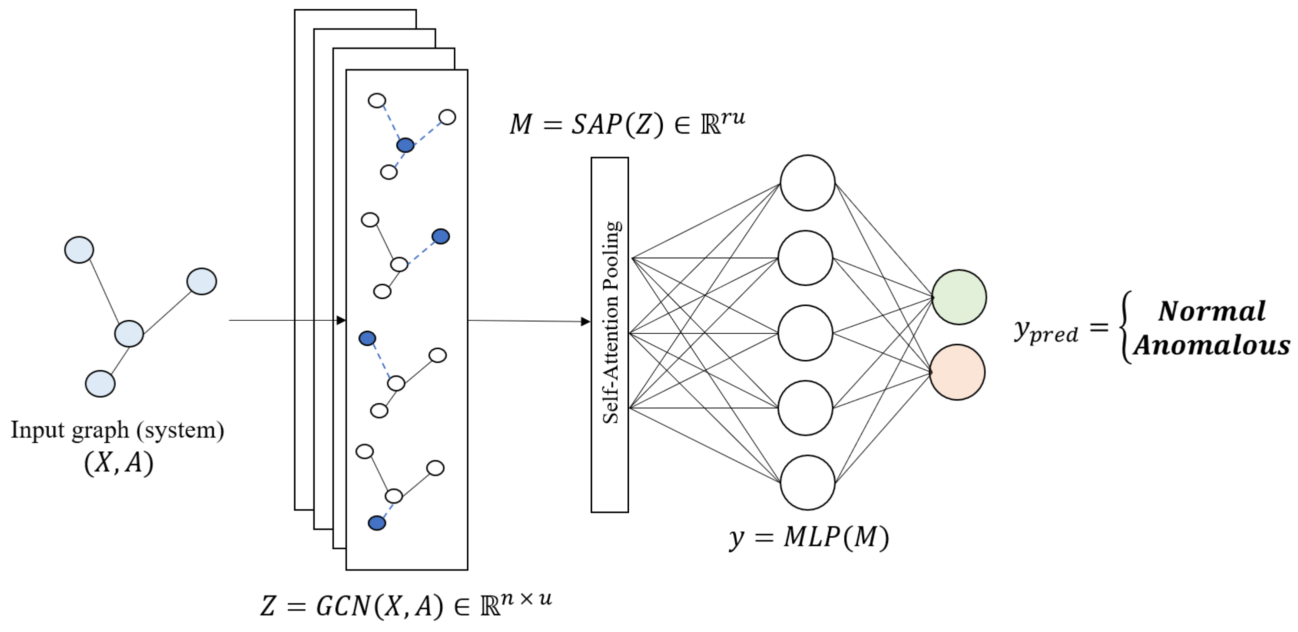
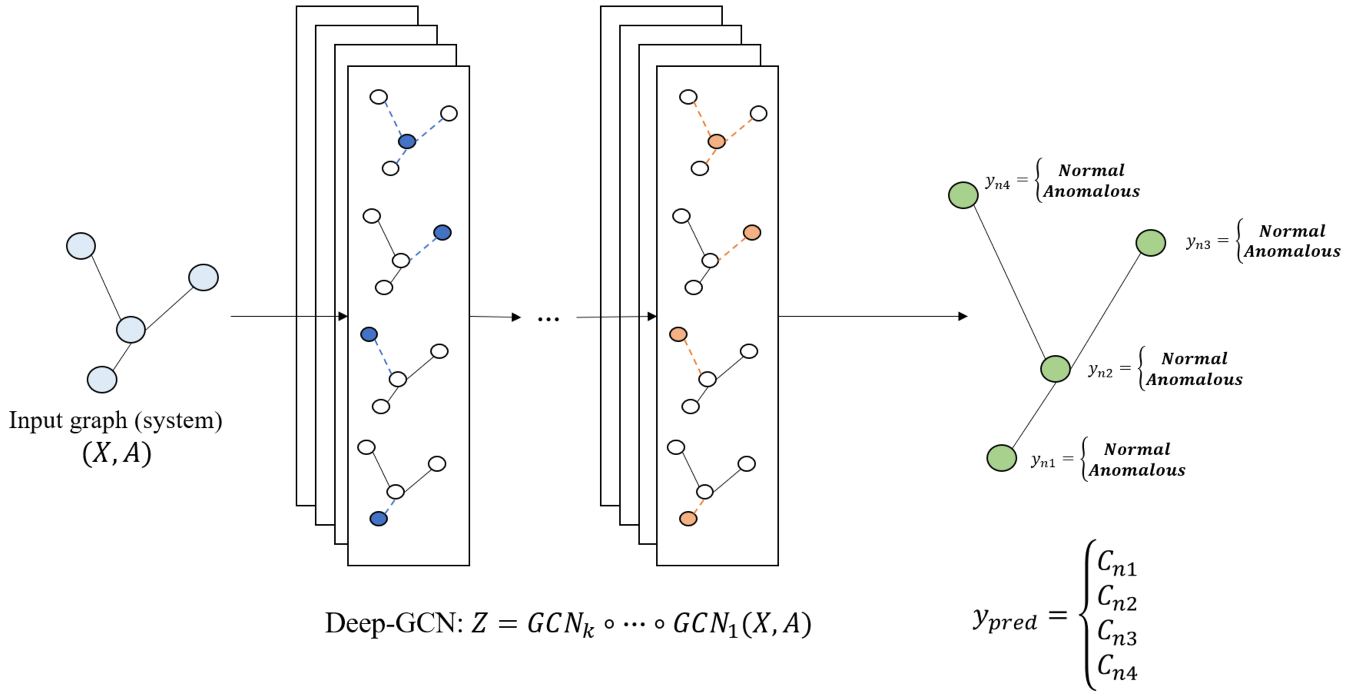

$M = SAP(Z) \in \mathbb{R}^{ru}$
Self-Attention Pooling
Input graph (system)
(X, A)
$y_{pred} = \begin{cases} \textbf{Normal} \\ \textbf{Anomalous} \end{cases}$
$y = MLP(M)$
$Z = GCN(X, A) \in \mathbb{R}^{n \times u}$

Figure 4.2: System-level module architecture.

4.3 Component-level module

As seen in Figure 4.1, if the system-level module detects an anomalous behavior for the system, one of the following steps is to use the **Component-level module** of the framework. In this module, a deep-GCN model (with k-layers) is used to perform a binary classification for each node of the graph that represents the studied system, which will correspond to the identification of a normal or anomalous behavior of each component in it ($C_{n\mathrm{i}} = \{Normal, Anomalous\}$, where ni corresponds to *no*de i). The architecture used in this module is presented in Figure 4.3.



Input graph (system)
(X, A)
...
Deep-GCN: $Z = GCN_k \circ \cdots \circ GCN_1(X, A)$
$y_{n4} = \begin{cases} Normal \\ Anomalous \end{cases}$
$y_{n3} = \begin{cases} Normal \\ Anomalous \end{cases}$
$y_{n2} = \begin{cases} Normal \\ Anomalous \end{cases}$
$y_{n1} = \begin{cases} Normal \\ Anomalous \end{cases}$
$y_{pred} = \begin{cases} C_{n1} \\ C_{n2} \\ C_{n3} \\ C_{n4} \end{cases}$

Figure 4.3: Component-level module architecture.

4.4 System components relationships module

Finally, as seen in Figure 4.1, if the system-level module detects an anomalous behavior for the system, the other following step is to use the **System components relationships module** of the framework. The aim of this module is to weight and visualize the relationships between the different components of the system when an anomalous behavior is present. To do this, an *Auto-Encoder* type of architecture is used, which is a type of model based on DL that tries to replicate and reconstruct its fed data in a unsupervised fashion, in this case the adjacency matrix of an input graph that represents the system in addition to information of sensors present in each of its components (nodes) in anomalous conditions. Given this, the following steps are performed:

1. A GCN model will be used to extract features directly from the graph information and structure (the pair (X, A). These features are going to be encoded in a latent

representation Z, so this first network is going to be referred as *Encoder*. This is going to be trained only with information from systems that exhibit an anomalous behavior with the aim of encoding relevant characteristics associated with this. Mathematically the encoder performs $Z = GCN(X, A)$.

2. Using the latent representation Z from the previous step, new information from systems that exhibits an anomalous behavior is going to be used to reconstruct its adjacency matrix A_{rec} using a sigmoid function, being this part referred as *Decoder*. A_{rec} is going to be reconstructed using the extracted features from the encoder output Z, so new relationships will "appear" between components that do not have direct connection in their prior adjacency matrix of the system, this is, no physical connections among them (this will be further explained in the next chapter). Mathematically the decoder performs $A_{rec} = \sigma(ZZ^T)$, where σ is the sigmoid function.

As seen in the previous steps, A_{rec} will be reconstructed using a sigmoid function, which will assign probabilities of existence for each reconstructed element in it [9][17]. Due to this, the use of Z will give more relevant information associated with an anomalous behavior of the system to the decoder, so new relationships will appear in A_{rec} different from the prior A. This "new" elements A_{rec} can be seen as implicit relationships between components of the system that are not directly connected by a link, this is, do not have any physical connection between them. Furthermore, existing elements will now correspond to a probability of existence, so they can be used to weight the relationships between components present in A_{rec}. To do the latter, a Min-Max scaling is used in the range $[0, 1]$, which will assign a weight of 1 to the largest element of A_{rec} and 0 to the lowest, representing the highest weight of relationship between components of the system and the lowest respectively. This scaled matrix is going to be named as *Weighted Relationships Matrix* (WRM) and its application in the proposed framework is going to be explained in the following section.

The architecture associated to this module is presented in Figure 4.4.

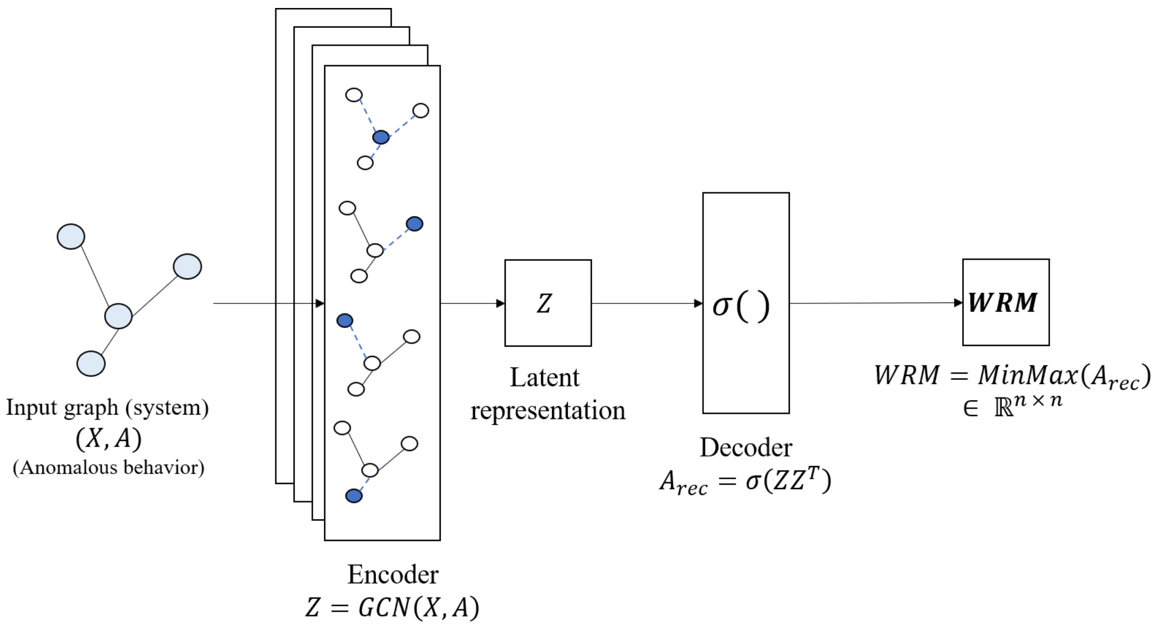


Figure 4.4: System components relationships module architecture.

4.5 Analysis regarding the outputs of the system-level PHM framework for health assessment

The aforementioned modules that compose the proposed framework for system-level PHM enables a deep analysis regarding the health assessment of a system. This can be done in three different levels: (1) system-level health assessment, (2) component-level health assessment and (3) analysis of the relationships between components of a system for anomalous scenarios. The three modules have as output a powerful tool for reliability engineers to do this, which are described as follows:

- **SLM output**: this module delivers the health state of the whole system, diagnosing the current behavior of it as *Normal* or *Anomalous*.
- **CLM output**: when the whole system behavior is diagnosed as *Anomalous*, it is of great relevance to know which components of it are responsible of this, so maintenance plans can be performed. Given this, this module delivers the health state of each component of the system, diagnosing the behavior of each as *Normal* or *Anomalous*.
- **SCRM output**: furthermore, as an addition on the previous point, this module give information on how the different components of the system are related to each other when the whole system exhibits an anomalous behavior. To do this, the WRM is delivered as an output of this module, where its elements represents the weight of relationship between the components of the system. A *MinMax scaler* was used on the decoder output to scale it in a way that its elements can be comparable among each other. It is important to notice that the WRM is a symmetrical matrix (due to the use of undirected graphs as a representation of the system), where, for its elements, 1 represents the highest relationship between components and 0 the lowest.

It is important to note that each of the previous modules work automatically, needing just the present time operational data of the system to work.

Chapter 5

Case Study: chlorine dioxide generation reboiler system in a cellulose plant

The following chapter describes the case study in which the proposed framework for system-level PHM is going to be tested and validated. A real-world industry problem is going to be addressed, making it a highly relevant problem to solve due to the multiple limitations that real-world datasets commonly exhibit.

5.1 Problem description

As a case study, sensor operational data from a chlorine dioxide generation re-boiler system in a cellulose plat is going to be used. This system, which is composed by multiple mechanical assets, has repeatedly problems on its reboiler system, so the aim of the proposed framework is to asses the health-state of it.

The reboiler system is mainly composed by five assets, which are listed and described above. Furthermore, the whole system is shown in Figure 5.2.

- **Generator (G)**: generates chlorine-dioxide (ClO_2) by a chemical reactions that needs as input ($NaClO_3$, CH_3OH, H_2SO_4) and water vapor heat. The main product of this reaction is condensed water, chlorine-dioxide (main product) and other not that relevant chemicals for this system, mainly composed by salts.
- **Reboiler (R)**: consists of a shell-and-tube heat exchanger that evaporates the condensed water from the generator and reinjects it again in it, controlling its level for the chemical process for ClO_2 production. Cold liquor runs through the tubes in a first step, being heated by the excess water vapor generated by the reboiler. Then, in a second step, the heated liquor is used to evaporate the condensed water coming from the generator.
- **Condensate tank (CT)**: the condensed water from the heating of the liquor in the reboiler is stored in this condensate tank, which can be then reinjected to the hot-water

system of the cellulose plant.

- **Hot-water system reinjection pump (M014)**: is directly connected to the condensate tank to control the reinjection of condensed water to the hot-water system of the cellulose plant.
- **Salts recuperation system reinjection pump (M015)**: is directly connected to the generator to control the flow of sub-product salts from the ClO_2 generation production to a salts recuperation system in the cellulose plant.

The main problem of the reboiler system is that, with time, sulphate inlays are generated in the interior of the reboiler tubes, degrading them and decreasing the generation of ClO_2, which will be considered as an anomalous behavior of the system. This inlays can be seen in Figure 5.1.

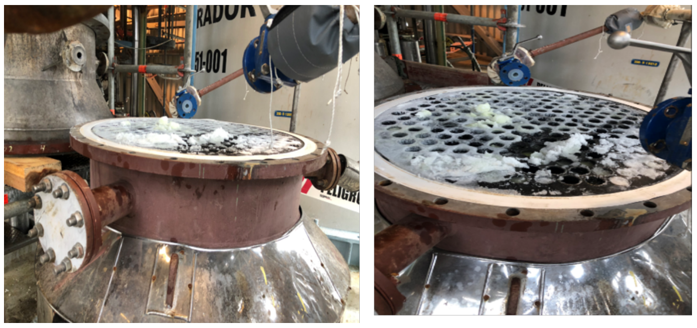


Figure 5.1: Sulphate inlays in the interior of the reboiler tubes.

To detect if the tubes of the reboiler are developing sulphate inlays, seven different operational sensors are used in the system and are described below. These can be seen in Figure 5.2.

- **Temperature sensor TIC8014** [$°C$]: measures the temperature of the water-vapor in the reboiler. High temperatures, above $110°C$ indicates an anomalous behavior.
- **Temperature sensor TI8015** [$°C$]: measures the temperature of the condensate water that goes from the reboiler to the condensate tank. High temperatures, above $110°C$ indicates an anomalous behavior.
- **Conductivity sensor CI8017** [$\mu S/cm$]: measures the conductivity of the condensed water that flows from the condensate tank to the M014 pump. Sudden conductivity increments indicates an anomalous behavior.
- **Pressure sensor PI8026** [kPa]: measures directly the pressure in the top of the generator. A sudden increase of pressure indicates an anomalous behavior.
- **Pressure sensor PIC8025** [kPa]: measures the differences of pressure in the generator from vacuum losses. A sudden increase of differences of pressure indicates an anomalous behavior.
- **Power of the M014 pump** [%]: measures the power load on the hot-water system reinjection pump. A high power load indicates an anomalous behavior.
- **Power of the M015 pump** [%]: measures the power load on the salts recuperation system reinjection pump. A high power load indicates an anomalous behavior.

As seen in Figure 5.2, there are pipelines with no sensor measurements present in the dataset that could have a repercussion on the results of the framework, specially for measurements associated to the power sensor of the M015 pump.

The data used as a case study for the proposed framework corresponds to the measurements of the aforementioned seven operational sensors from October of 2008 to September of 2018 with a sampling frequency of 2 $[hr]$.

5.2 Data exploration and pre-processing

As mentioned in the beginning of this chapter, given that the case study data used in this work comes from a real-world industry, its structure is not ideal a priori, showing some limitations to be directly used to train and validate the proposed framework. Due to this, a data exploration step is needed to then pre-process data.

The dataset is composed by 42.206 measurements through time for the 7 sensors (i.e. a total of 295.442 data points). Some prior labels where given by the cellulose plant reliability engineers, consisting of a total of 1.820 labels that indicates if at a given time the tubes of the reboiler exhibits sulphate inlays. Also, some sensor measurements were NaN valued (not a number) and the CI8017 sensor showed miss-readings that indicates values of order 10^5. Given this, the first step of the pre-processing was to delete all the sensor measurements that exhibit these limitations, which corresponds to 667 time measurements (i.e 4.669 data points or 1.58 [%] of the dataset). After this, 41.539 measurements were left to be worked with the framework.

As pointed above, only 1.820 labels were given by the cellulose plant reliability engineers, so to label the full dataset a semi-supervised ML-based procedure needed to be made. This was performed by the following steps:

1. The data was sensor-wise scaled using an Standard Scaling, so data points with high standard deviation can be detected easily.
2. A PCA model was used on the scaled data, reducing the seven prior sensors to only three principal components that summarize its variance. This was done with the aim of feeding this transformed data to a unsupervised clustering algorithm, easing the process with less but highly relevant data.
3. the PCA transformed data was fed to a DBSCAN to perform a clustering analysis on it, so one cluster was detected on data as a normal behavior of the system. The detected noise from the algorithm will correspond to data points that indicate an anomalous behavior of the system.

To fine-tune the DBSCAN parameters, a semi-supervised iterative approach was taken. A first set of parameters (ε, S) (see Chapter 3) were used, comparing the DBSCAN detected cluster accuracy with respect to the prior-known labels of the dataset. Iterating over these parameters, the final tuned values were $\varepsilon = 0.4$ and $S = 700$, which delivers a clustering with an accuracy of 88 [%] for true positives (i.e. correctly detected normal behaviors of the

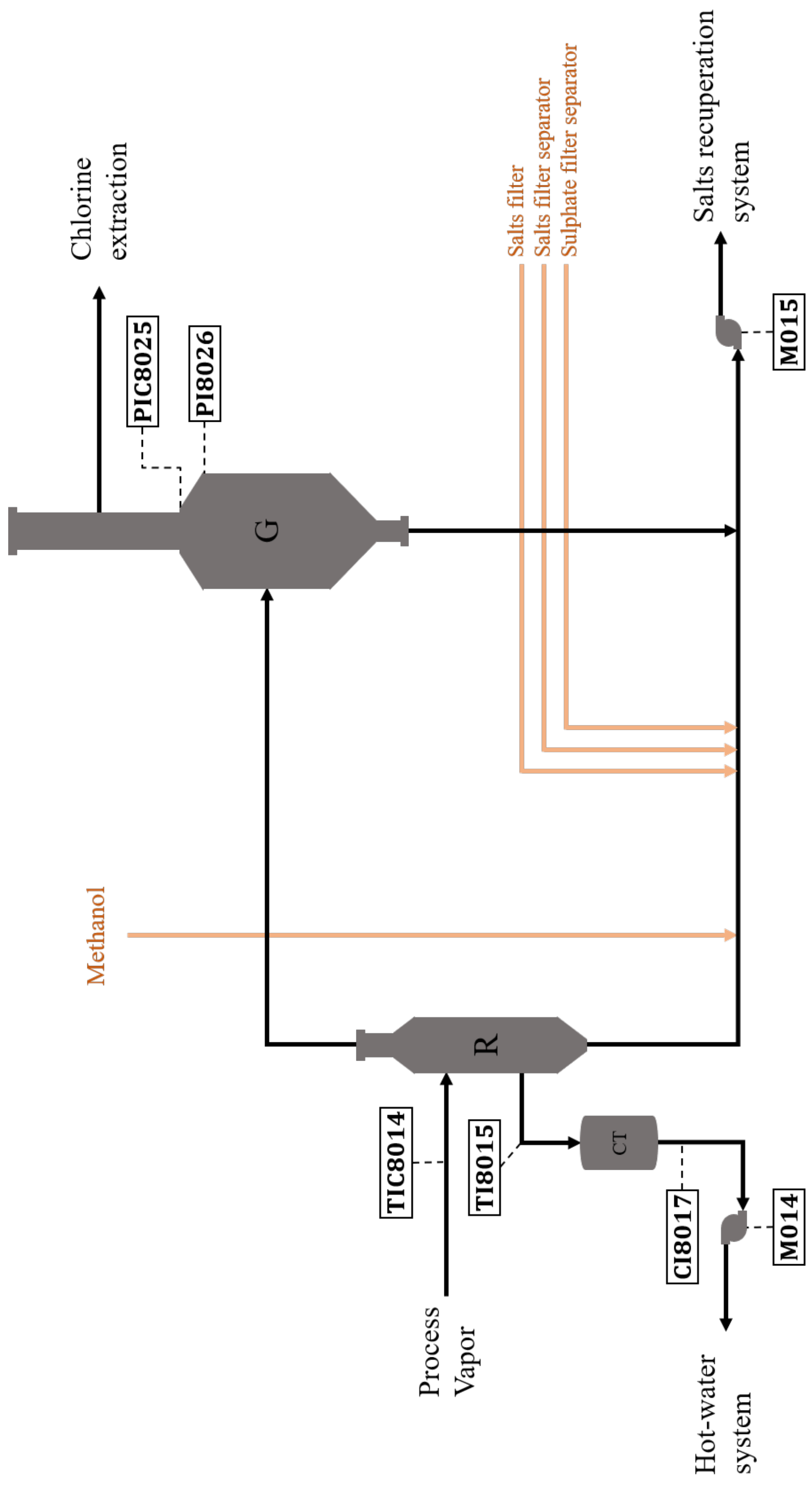


Figure 5.2: Chlorine dioxide generation re-boiler system. In black: physical pipeline and flow direction that connects each component. In orange: other pipelines present in the system but with no sensor measurements

system) and 100 [%] for true negatives (i.e. correctly detected anomalous behaviors of the system). Using this, the whole dataset was labeled. The detected cluster can be visualized on the PCA-transformed dataset in Figure 5.3.

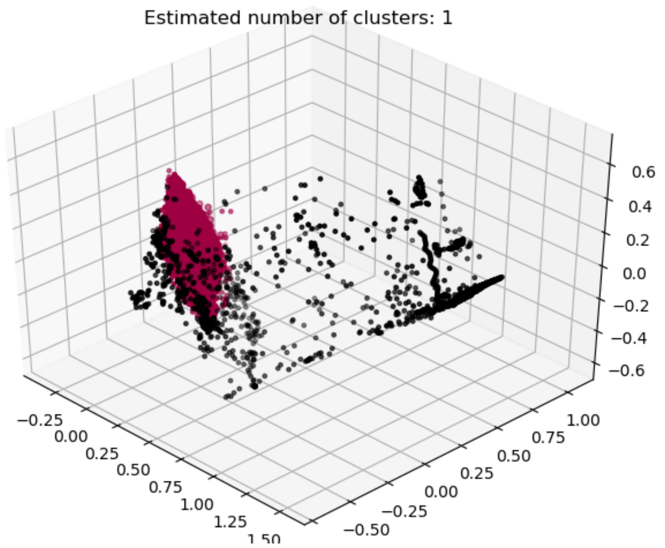


Figure 5.3: PCA transformed data DBSCAN clustering. In burgundy: detected cluster, i.e. normal system behavior data points. In black: detected noise, i.e. anomalous system behavior data points.

As it can be seen, the previous labeling procedure works on the system-level, this is, labels will indicate only if the whole system is behaving normally or not, which is only useful of the system-level module of the proposed framework. Given this, hand-crafted labels must be made to be used by the components-level module, due to the lack of prior-known labels for each of the mechanical assets (i.e. components) that compose the reboiler system. To craft these, the minimum and maximum values of each sensor on both the prior-known and DBSCAN-clustered normal behavior labels are used: lets name these as the prior-known threshold (PT) and DBSCAN-clustering threshold (CT). A definitive threshold is built to consider a component as behaving normally, where the inferior (inf) and superior (sup) values of it corresponds to $inf = \min(PT, CT)$ and $sup = \max(PT, CT)$ respectively. Given this, the final threshold can be seen in Table 5.1.

Given the final threshold for each operational sensor, a hand-crafted label is assigned for each component of the system. If at least one of the sensors that measures data related to a component is out of it, the component will be labeled as with *Anomalous behavior*, else, it is considered as with *Normal behavior*.

Finally, for the whole dataset, the class-balance on the **System-level** labels is 82/18 [%] for normal and anomalous health state respectively. On the other hand, for the **Component-level** labels, the balance is 57/43 [%] for the *Reboiler*, 73/27 [%] for the *Condensate tank*, 62/38 [%] for the *M014 pump*, 96/4 [%] for the *Generator* and 68/32 [%] for the *M015 pump*, giving a balance of 71/29 [%] for the whole component labels.

	Normal behavior thresholds - prior-known labels						
	TIC8014 [C]	TI8015 [C]	CI8017 [$\mu S/cm$]	PI8026 [kPa]	PIC8025 [kPa]	M014 [%]	M015 [%]
inf	97,09	75,14	6,79	13,11	12,98	63,46	37,55
sup	102,41	84,15	9,63	27,44	27,33	78,54	49,04

	Normal behavior thresholds - Detected cluster						
	TIC8014 [C]	TI8015 [C]	CI8017 [$\mu S/cm$]	PI8026 [kPa]	PIC8025 [kPa]	M014 [%]	M015 [%]
inf	99,32	75,76	1,16	17,89	17,95	42,48	37,54
sup	105,76	86,03	6,79	20,70	20,46	78,49	46,96

	Final threshold						
	TIC8014 [C]	TI8015 [C]	CI8017 [$\mu S/cm$]	PI8026 [kPa]	PIC8025 [kPa]	M014 [%]	M015 [%]
inf	97,09	75,14	1,16	13,11	12,98	42,48	37,54
sup	105,76	86,03	9,63	27,44	27,33	78,54	49,04

Table 5.1: Normal behavior threshold for each operational sensor for the prior-known labels, the detected DBSCAN cluster and the final one used for labeling the components of the system.

5.3 Modeling of the system as a graph

Having done the pre-processing of the dataset, the next step is to represent the system shown in Figure 5.2 as an undirected graph. To do this, the mechanical assets of the reboiler system will be considered as *nodes* of the graph and physical connections will be considered as the *links*, which in this case correspond to the connecting piping lines. Finally, a "snowball sampling" technique is used to construct the graph, using as *seed* node the Reboiler. The features associated to each node will correspond to the operational sensors, which can be directly measuring on a component or in the pipe between two components. For the latter, the sensor will be assigned as a feature for both of them. Given this, the reboiler system can be represented as shown in Figure 5.4. The following tag will be used for each node of the graph:

- **Node 0**: Reboiler.
- **Node 1**: Condensate tank.
- **Node 2**: M014 pump.
- **Node 3**: Generator.
- **Node 4**: M015 pump.

The *Adjacency Matrix* $\hat{A}$ associated to the system graph is given by,

$$\hat{A} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{5.1}$$

which is symmetrical due to its undirected graph nature. On the other hand, the *Feature Matrix* X will contain the sensor values S_{i} with i = $\{TIC8014, TI8015, CI8017, PI8026,$

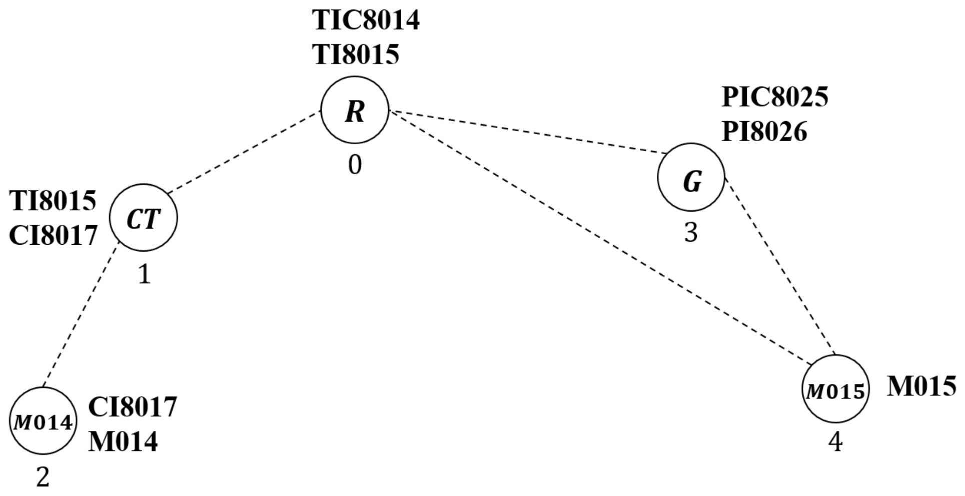


Figure 5.4: Reboiler system represented as graph. Each node corresponds to a component of the system with their associated node features in bold.

$PIC8025, M014, M015\}$ for each of the nodes in the rows.

$$X = \begin{pmatrix} S_{TIC8014} & S_{TI8015} & 0 & 0 & 0 & 0 & 0 \\ 0 & S_{TI8015} & S_{CI8017} & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{CI8017} & S_{M014} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{PI8026} & S_{PIC8025} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{M015} \end{pmatrix} \tag{5.2}$$

5.4 Dataset construction to feed the framework

The data used as case study for the proposed framework comes as a *.csv* with 42.206 rows and 7 columns corresponding to time and sensor measurements respectively. After pre-processing, only 41.539 rows are left, with each column (sensor) scaled in the range $(0, 1)$ using a MinMax Scaler. For each time-measurement a pair (A, X) is generated according to equations 5.1 and 5.2. This pair is going to represent the independent variable of the dataset. On the other hand, the dependent variable Y will be constructed according to the module of the framework:

- **SLM**: for each pair $(A, X)_\text{i}$ a label $y_{SLM} = C_\text{i}$ with i $= \{1, ..., 41.539\}$ is generated. C_i will correspond to the health state of the system, which is encoded as *0: Normal* and *1: Anomalous.*
- **CLM**: for each pair $(A, X)_\text{i}$ a vector-shape label $y_{CLM} = C_{\text{i},j} \in \mathbb{R}^{1 \times N_c}$ with i $= \{1, ..., 41.539\}$, N_c the number of components of the system (i.e. 5) and $j = \{1, ..., N_c\}$ is generated. $C_{\text{i},j}$ will correspond to the health state of each component j of the system in the time-measurement i, which is encoded as *0: Normal* and *1: Anomalous.* This is the equivalent of having a single binary label for each system component.
- **SCRM**: no dependent variable is computed due to its unsupervised nature.

Finally, the dataset is splitted into a training and testing set randomly in a ratio of 80/20 [%]. These are going to be used to train and validate each module respectively.

5.5 System-level PHM framework hyperparameter tuning and performance metrics

For each module that compose the framework, an extensive hyperparameter search was done to find the optimal ones to deliver the beast results. Each of them are going to be shown in the following sections.

5.5.1 System-level module

SLM training hyperparameters

- **Epochs**: 700.
- **Learning rate**: 0.0001.
- **Loss function**: Softmax-Crossentropy.
- **Optimizer**: Adam.
- **Batch size**: 64.

SLM model architecture hyperparameters

Graph Convolutional Network (GCN)

- **Number of layers**: 1.
- **Feature maps**: 32.
- **Activation functions**: SELU.
- **Regularizers**: $l_1(s = 0.1)$.

Self-Attention Pooling (SAP)

- W_{s1} **neurons**: 16.
- W_{s2} **neurons**: 5.

Muti-layer Perceptron (MLP)

- **Number of layers**: 4.
- **neurons**: [64, 32, 8, 2].

- **Activation functions**: [SELU, SELU, SELU, Softmax].
- **Regularizers**: [$l_1(s = 0.1)$, $l_2(s = 0.01)$, $l_2(s = 0.01)$, None].

5.6 Component-level module

CLM training hyperparameters

- **Epochs**: 1.000.
- **Learning rate**: 0.01.
- **Loss function**: Softmax-crossentropy.
- **Optimizer**: Adam.
- **Batch size**: 32.

CLM model architecture hyperparameters

- **Number of layers**: 5.
- **Feature maps**: [16, 16, 16, 16, 2].
- **Activation functions**: [SELU, SELU, SELU, SELU, Softmax].
- **Regularizers**: None.

5.6.1 System components relationships module

SCRM training hyperparameters

- **Epochs**: 3.500.
- **Learning rate**: 0.001.
- **Loss function**: Root Mean Squared Error (RMSE).
- **Optimizer**: Adam.
- **Batch size**: 32.

SCRM model architecture hyperparameters

Encoder

- **Number of layers**: 2.
- **Feature maps**: [16, 16].
- **Activation functions**: [SELU, SELU].
- **Regularizers**: None.

5.6.2 Performance metrics

To evaluate the diagnosis performance for heath assessment of both the SLM and CLM of the proposed framework, the following metrics are going to be used: confusion matrix and precision-recall F1, Area Under the Curve (AUC) and Average Precision (AP) scores. These are used due to the high imbalance between normal and anomalous behaviour for both system and component level health assessments. Furthermore, the overall accuracy (i.e. *Correct predictions / Total predictions*) is not used given that it can be misleading to evaluate the performance of the framework.

Confusion Matrix

A confusion matrix (CM) is a square table layout that allows the visualization of the performance of a classification algorithm. The number of rows/columns corresponds to the different classes to predict, in this case, *Normal* and *Anomalous* behavior of a system or component. The rows represent the predicted values and the columns the ground-truth ones. Inside the CM, the following values are shown (see Figure 5.5):

- **True Positives (TP)**: Normal behaviors that are correctly predicted.
- **False Positives (FP)**: Normal behaviors that are predicted as Anomalies.
- **False Negatives (FN)**: Anomalous behaviors that are predicted as Normal.
- **True Negatives (TN)**: Anomalous behaviors that are correctly predicted.

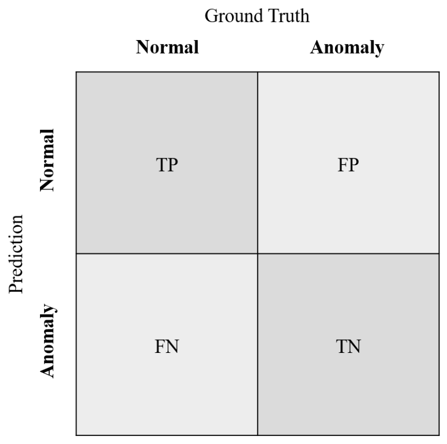


Figure 5.5: Confusion Matrix.

For the SLM, a CM is going to be given for the whole dataset. For the CLM, a CM is going to be generated for the overall accuracy of the model and for each component health-state prediction in the system.

Precision-Recall scores

For the SLM and CLM, Precision (P) -Recall (R) F1, AUC and AP scores ($P - R$ scorees) are going to be computed given that the used dataset health state labels are imbalanced. Saito et. al [27] showed that these metrics enables a good description of the classification performance of a model trained with imbalanced datasets.

Precision and Recall are defined as follows:

$$P = Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$R = Recall = \frac{TP}{TP + FN} \tag{5.4}$$

where a high *Precision (P)* and *Recall (R)* indicates higher TPs with lower FNs and lower FPs with higher TNs respectively.

Computing the output of the module as the probability of existence of a health state (which is done by the sotfmax activation function at the end of each module), it is important to define a threshold by which a predicted class will exist. Is a common practice in literature to set this threshold at $y > 0.5$, which will be also used in this work. To $P - R$ scores for the model, 1400 different thresholds between 0 and 1 are going to be used.

Using P and R the *skill* of a model can be analyzed, which is highly relevant to class imbalanced datasets. A skillful model will, on average, give a higher probability output to a random chosen true occurrence (i.e. TP or TN) than a false one (i.e. FP or FN).

F1 score

Corresponds to the harmonic mean of the P and R. This summarized the model skill for the specified probability threshold. Ranging from 0 to 1, a higher value shows better performance of the model.

$$F1 = 2\frac{PR}{P + R} \tag{5.5}$$

Area under the curve (AUC) score

Corresponds to the area under the plot $P = P(R)$ as calculated as:

$$AUC = \int_0^1 P(R)\mathrm{d}R \tag{5.6}$$

This score summarizes the skill of the model across a set of probability thresholds. Ranging from 0 to 1, a higher value shows better performance of the model.

Average precision (AP) score

Considering k thresholds to consider a certain class as existent or not (e.g. $0.4 \leq y$ instead of $0.5 < y$), the AP score is given by:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{5.7}$$

where P_n and R_n are the precision and recall at the n-th threshold. This score summarizes the weighted increase in precision with each change in R for a set of probability thresholds. Ranging from 0 to 1, a higher value shows better performance of the model.

Chapter 6

Results and Discussion

In this chapter the results concerning the performance of the proposed framework for system-level PHM are shown. To validate the use of graph convolutions as feature extractors in each module, the proposed models will be compared in terms of confusion matrix and precision-recall scores (summary for 10 runs of each model) with a simple MLP model (i.e. comparison to another DL model) and a Random Forest (i.e. comparison to a traditional ML model). For the case of the SCRM, there are no similar approaches in literature to compare with, so just the obtained results and related analysis is presented.

The MLP model used for comparison with the SLM results uses the same hyperparameters that the one in the SLM model (without the GCN and AP feature extraction. See section 6.1.2). On the other hand, the MLP used for each component of the system to compare it to the CLM was optimized by using a GridSearch, having finally two 16-neurons ReLU activated layers batch-wise trained using an Adam optimizer with learning rate of 0.01 for 100 epochs and batches of data of 64 random samples.

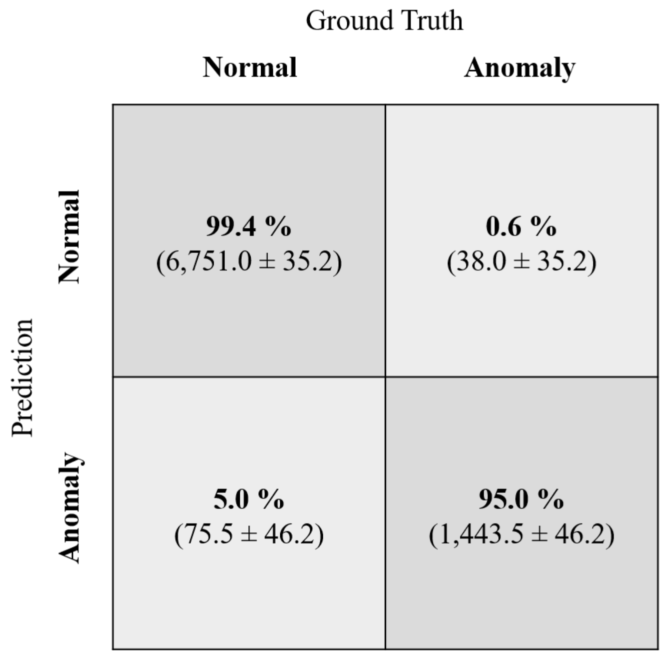
The Random Forest model used for comparison on both the SLM and each component of the CLM had optimized parameters after a GridSearch with 3-fold cross validation, with total of 330 possible parameter combinations. The model uses 100 estimators, *Gini* separation criterion, 2 minimum sample leafs and 1 minimum sample split.

For the industry, it is much more important to detect anomalies in their systems, which will help them to avoid the danger and repair costs associated to them. Given this, true negatives, (i.e. accurate predictions of anomalous behaviors) are prioritized over TPs, FPs and FNs in the CM. Furthermore, due to the unbalanced nature of the health state labels in the dataset (see Chapter 5), a highly skillful framework is essential, this is, high $P - R$ scores are expected.

Lastly, before presenting the obtained results, it is important to state that the proposed framework and associated modules were developed with no hand-tuned dataset specific parameters such as label specific-weights or classification criterion different to $y > 0.5$. The main idea of this work is to explore the capabilities of the use of graph neural network based models on system-level PHM, so no bias regarding an overfit to the studied case study dataset is desired, even though some results are not optimal at first glance.

6.1 System-level module results

The CM associated to the proposed SLM is presented in figure 6.1. As it can be seen, the proposed module presents a high accuracy for both TPs (99.4 [%]) and TNs (95.0 [%]) with low standard deviations, which is remarkable given that the dataset was not hand-balanced.



Prediction \ Ground Truth	Normal	Anomaly
Normal	**99.4 %** (6,751.0 ± 35.2)	**0.6 %** (38.0 ± 35.2)
Anomaly	**5.0 %** (75.5 ± 46.2)	**95.0 %** (1,443.5 ± 46.2)

Figure 6.1: SLM Confusion Matrix.

The comparison of the CM results with respect to an MLP and Random Forest models are presented in table 6.1. As it can be seen, the overall performance of the proposed model outperforms the MLP and Random Forest, with the exception of the TP rate associated to the MLP model, which is 0.5 [%] higher. The latter is not relevant comparing the overall performance of the models, as it can be seen that the MLP overfits to the most common class (i.e. normal behavior of the system), which leads to a much worse TN rate. Furthermore, is it possible to see that the feature extraction performed by the graph convolutional layers (and embedded by the self-attention pooling) is highly relevant to a DL model performance, this is, giving explicitly the physical relationships among the system components, avoiding the overfit present on the simple MLP to the normal health state label. Moreover, it is important to notice that the Random Forest also shows a good overall performance, but, as it will be seen in the following section (CLM results), this model can not be used in the same fashion as a GCN, thus not enabling the training of just one model to perform component-level predictions.

Regarding the $P-R$ scores, table 6.4 shows the results of the proposed SLM, MLP and Random Forest models. As it can be seen, the proposed SLM architecture outperforms the

	proposed SLM				MLP				Random Forest			
	TP[%]	FP[%]	FN[%]	TN[%]	TP[%]	FP[%]	FN[%]	TN[%]	TP[%]	FP[%]	FN[%]	TN[%]
Mean	**99.4**	**0.6**	$\underline{\textbf{5.0}}$	$\underline{\textbf{95.0}}$	$\underline{99.9}$	$\underline{0.1}$	64.5	35.5	99.5	0.5	6.1	93.9
std	0.5	0.5	3.0	3.0	0.2	0.2	9.2	9.2	0.1	0.1	0.3	0.3

Table 6.1: Comparison of the proposed SLM model, MLP and RF confusion matrices for system-level health assessment. In bold: proposed model results. Underlined: best performance values

other models with respect to the F1, AUC and AP scores. Furthermore, this three scores are very high valued, which shows that the proposed architecture is highly skillful (see section 6.4.2) and accurate for system-level health assessment. Also, it is important to notice how the addition of the graph convolutions imply a high performance and skill-boost with respect to the simple MLP model with no previous GCN feature extraction.

	proposed SLM			MLP			Random Forest		
	F1	AUC	AP	F1	AUC	AP	F1	AUC	AP
Mean	$\underline{\textbf{0.962}}$	$\underline{\textbf{0.996}}$	$\underline{\textbf{0.996}}$	0.322	0.526	0.526	0.957	0.992	0.990
std	0.011	0.001	0.001	0.000	0.000	0.000	0.002	0.001	0.001

Table 6.2: Comparison of the proposed SLM model, MLP and RF precision-recall scores for system-level health assessment. In bold: proposed model results. Underlined: best performance values

6.2 Component-level module results

The overall CM of the CLM results is presented in figure 6.5, where it can be see a high accuracy of the module, predicting a total of 94.1 [%] TPs and 89.7 [%] TNs correctly. It is important to mention that a single CLM architecture is trained to predict the health state of all individual components of the system, a characteristic enabled by the use of a GCN-based model. On the other hand, for the MLP and Random Forest models, each one needs to be trained individually for each component, which is the common practice nowadays when performing PHM in systems, an approach that needs to change to perform a better and more complete health assessment [1].

The CM related results concerning each particular component is shown in figure 6.6. From this figure it can be seen that, in general, the overall accuracy rate for TPs and TNs is high (over 97.0 [%]), with the exception of the TP rate of the Generator and the TN rate of the M015 pump, which are 82.5 [%] and 61.5 [%] respectively (see sub-figures (d) and (e) in the figure). Furthermore, the $P - R$ scores of each of component is presented in table 6.4, showing in general an skillful model regarding F1, AUC and AP scores with exception of the Generator and M015 pump low-valued F1 scores.

Regarding the results obtained on both the Generator and the M015 pump, a high standard deviation can be seen in its CM (see figure 6.6 (d) and (e)) positives (TP, FP) and negatives (FN, TN) rates respectively. To explain this, it is of high relevance the particular values of their F1 scores (of 0.365 and 0.673 respectively), which indicates a low-skillfulness of the model to predict true rates (i.e. TP and TN) for the used prediction threshold of $y > 0.5$. On the other hand, both components of the system show high AUC and AP scores as presented in table 6.4, which means that for other thresholds, the model can show more skill. Given this, the high standard deviation of the CLM model to predict true rates for the commonly used threshold of $y > 0.5$ in this components must be due to the lack of information variability on the sensors associated, so the difference between the two health states is not easily grabbed in the feature extraction step of the training process. To show this, the cumulative distribution function of the PIC8025 and PI8026 sensors associated to the generator and the M015 power sensor associated to the M015 pump are computed and showed in figures 6.2 and 6.3.

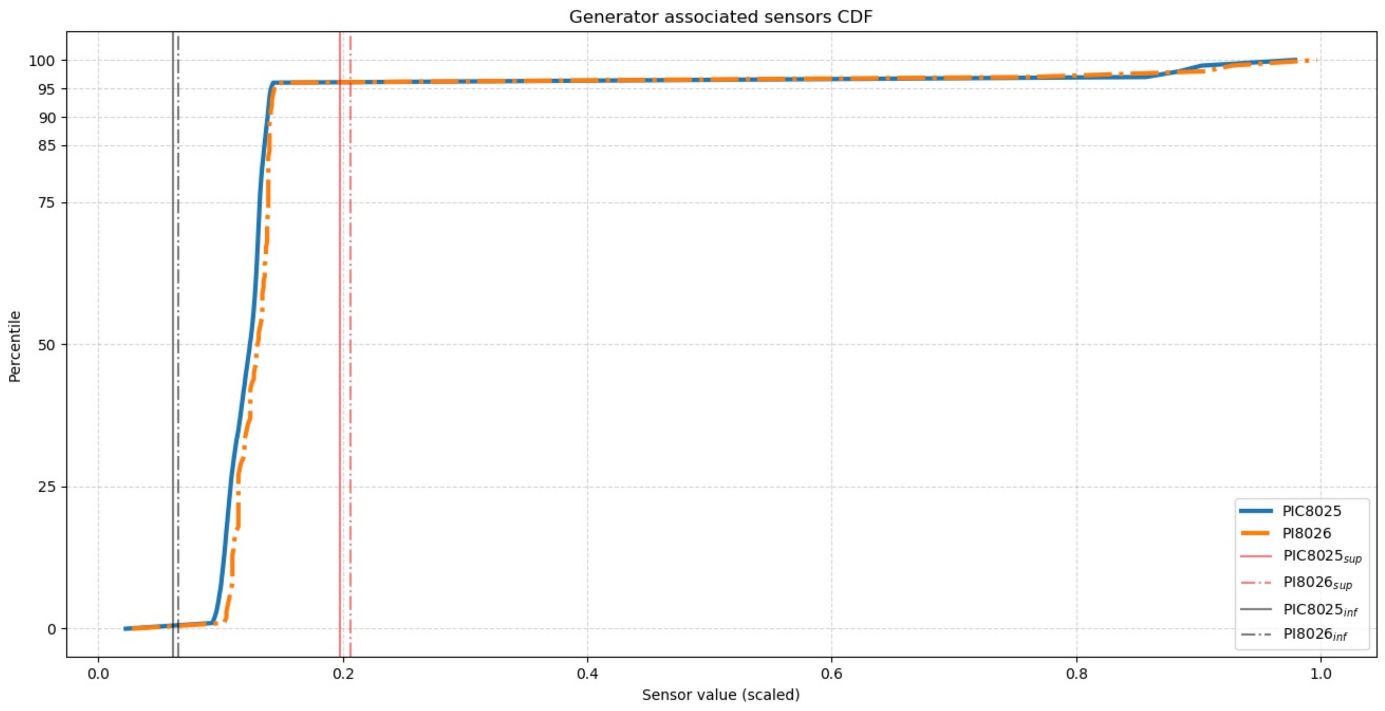


Figure 6.2: Generator associated sensors PIC8025 and PI8026 CDFs. Subscripts *inf* and *sup* in the plot legend refer to the inferior and superior values respectively of the normal behavior defined threshold from table 5.1

Regarding the generator, as it can be seen in figure 6.2, sensors PIC8025 and PI8026 have almost no difference, making the use of the two of them practically redundant to extract features for health state diagnosis. The main problem of this is that, even though two sensors are monitoring this component, there is no measurement variability, this is, less information is present for the model to extract features from it. The same phenomenon can be seen in the M015 pump from the figure 6.3, where only one sensor is used for monitoring, which reduces the capabilities of the model for feature extraction.

As a comparison to justify the previous analysis, the cdf of the reboiler associated sensors TIC8014 and TI8015 is computed and showed in figure 6.4, due to its good CM and $P - R$ scores results. Here it can be seen explicitly that there is much more monitoring variability from these two sensors, which means that more information can be used by the model feature

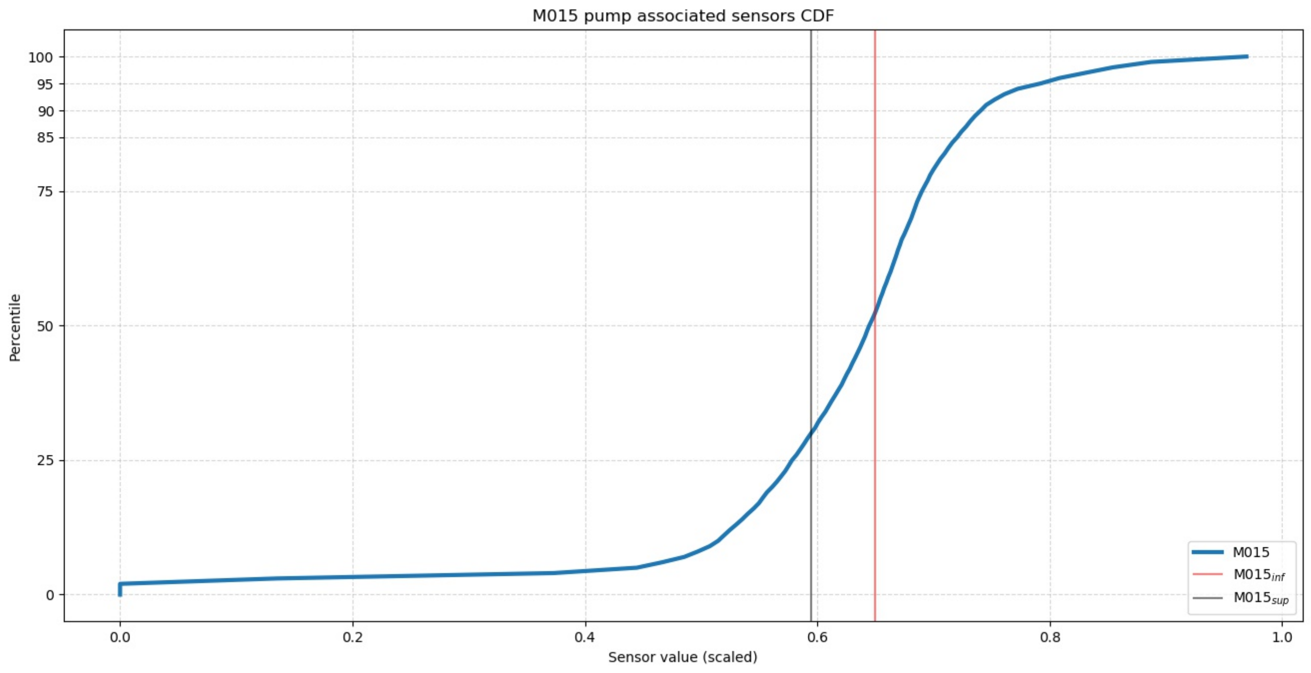


Figure 6.3: M015 pump associated sensor M015 CDF. Subscripts *inf* and *sup* in the plot legend refer to the inferior and superior values respectively of the normal behavior defined threshold from table 5.1

extraction training process. Furthermore, its is important to notice that, unlike the generator and M015 pump sensors, the reboiler sensors are located in the pipeline that connects it to different parts of the system as it can be seen in figure 5.2, a characteristic that is also present in the condensate tank and the M014 pump, which also deliver good results. On the other hand, the monitoring of the generator and the M015 pump is done directly on them. Two insights can arise from this phenomenon: Firstly, a much better health state diagnosis performance is achieved by the model when sensor information from the pipelines that connects the components is delivered, showing that the health state of each one of them is dependant to the others to which they are connected in the system. Secondly, using the latter insight as starting point, the CLM shows that, for a much valuable monitoring of both the generator and the M015 pump (in terms of health assessment), sensors must be placed in the pipelines that connects them to each other and to the generator (see figure 5.2 as reference).

Regarding the positive rates of the CM associated to the generator (see 6.6 (d)), a high standard deviation can be seen. The reason of this can be interpreted from figure 6.2. Here, for both PIC8025 and PI8026 sensors, it can be seen that the normal behavior threshold include almost all measurements from them with a steep slope but with an abrupt change near the inferior and superior limits of it that stays almost constant. This means that data inside the normal behavior threshold is, in general, almost the same in value, but with some measurements that can be easily miscalassified by the model due to their similarity to the anomalous behavior data, which explains the high standard deviation regarding the

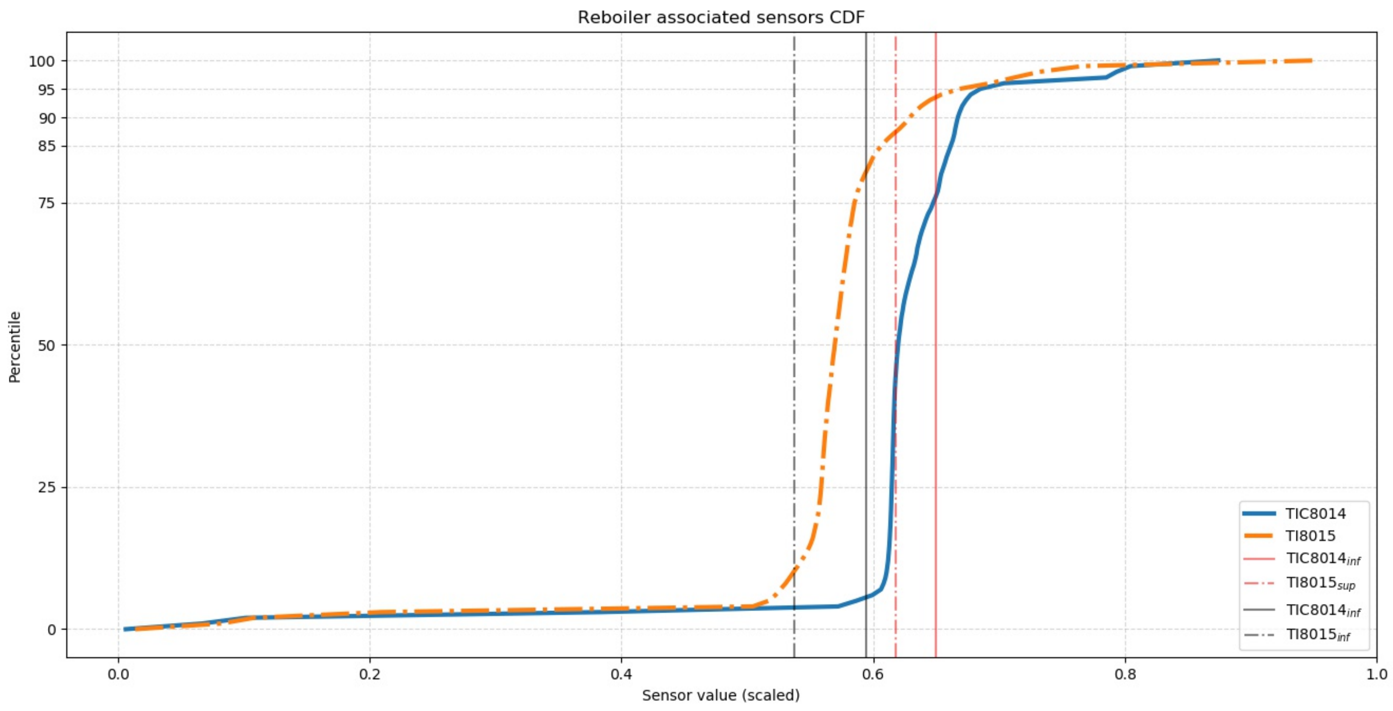


Figure 6.4: Reboiler associated sensors TIC8014 and TI8015 CDFs. Subscripts *inf* and *sup* in the plot legend refer to the inferior and superior values respectively of the normal behavior defined threshold from table 5.1

classification of normal behavior states in the positive rates of it CM (i.e. TPs and FPs). On the other hand, the M015 pump shows an opposite behavior in its CDF shown in figure 6.3. Here it can be seen a much more soft curve, with values inside the normal behavior threshold that shows clear difference between each of them. Furthermore, data outside the threshold (i.e. anomalous) maintains these behavior, showing almost no difference in slope with normal behavior data. Given this, the model can easily missclassify anomalous data near the normal threshold as part of it, which can be seen by the high standard deviation present in the negative rates of its CM (i.e. FNs and TNs).

Looking the CM results for the simple MLP and Random Forest models in table 6.3 it is clear that the proposed CLM architecture has a better overall accuracy regarding TPs and TNs, this is, predicting the health state of each component. In general, the worst results are associated with the simple MLP, presenting a disappointing performance principally in the predictions of normal health behaviors of the components. This shows that the use of GCNs can be a promising tool to tackle system-level big data problems using DL. On the other hand, the Random Forest model showed excellent results regarding the reboiler, condensate tank and generator, but unsatisfactory ones on the M014 and M015 pumps, presenting highly inaccurate rates of TNs in both. Furthermore, it is important to see that the three models struggles performing health diagnosis on the M015 pump, which can be explained by the analysis presented previously in this section. Also, from figure 5.2, it can be seen that other pipelines can affect the power output measurements (from sensor M015) in the M015

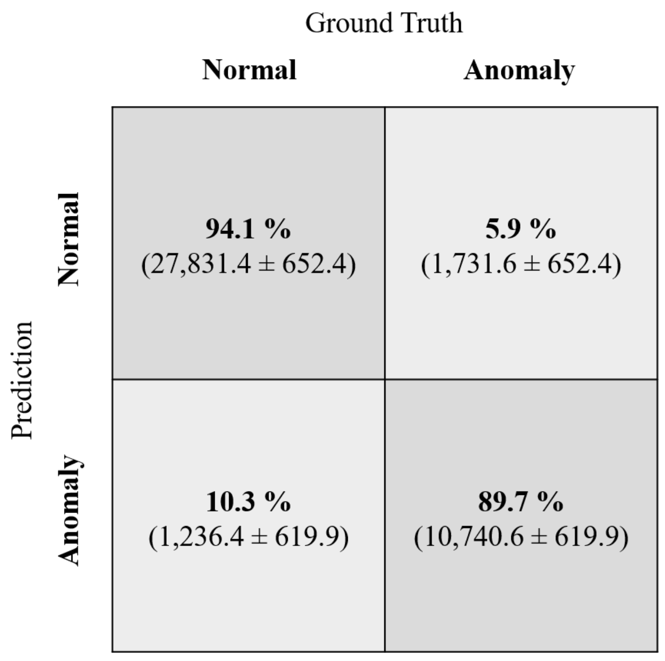


Figure 6.5: CLM Confusion Matrix for all available components labels.

		proposed CLM				**MLP**				**Random Forest**			
		TP [%]	FP [%]	FN [%]	TN [%]	TP [%]	FP [%]	FN [%]	TN [%]	TP [%]	FP [%]	FN [%]	TN [%]
Reboiler	Mean	**97.5**	**2.5**	**2.5**	**97.5**	9.9	90.1	0.1	99.9	100.0	0.0	0.0	100.0
(node 0)	std	0.5	0.5	1.2	1.2	29.7	29.7	0.1	0.1	0.0	0.0	0.0	0.0
Condensate tank	Mean	**98.7**	**1.3**	**3.0**	**97.0**	0.0	100.0	0.0	100.0	100.0	0.0	0.0	100.0
(node 1)	std	0.7	0.7	0.8	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
M014 pump	Mean	**97.9**	**2.1**	**1.0**	**99.0**	0.0	100.0	0.0	100.0	84.2	15.8	37.4	62.6
(node 2)	std	0.6	0.6	0.8	0.8	0.0	0.0	0.0	0.0	0.3	0.3	0.5	0.5
Generator	Mean	**82.5**	**17.5**	**2.3**	**97.7**	0.0	100.0	0.0	100.0	100.0	0.0	0.0	100.0
(node 3)	std	7.9	7.9	1.2	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
M015 pump	Mean	**99.5**	**0.5**	**38.5**	**61.5**	95.1	4.9	74.8	25.2	76.7	23.3	59.3	40.7
(node 4)	std	0.4	0.4	22.7	22.7	0.7	0.7	1.4	1.4	0.2	0.2	0.3	0.3

Table 6.3: Comparison of the proposed CLM model, MLP and RF confusion matrices for component-level health assessment. In bold: proposed model results. Underlined: best performance values

pump (presented in orange in the figure), which are not directly monitored by sensors in this pipeline.

Finally, regarding $P-R$ scores of the simple MLP and Random Forest models (see table 6.4), the proposed CLM architecture shows to be a much more skillful model in general, with the exception to the F1 score associated to the generator. For The MLP models, again, shows unsatisfactory scores, showing to be an approach that not only delivers bad classification accuracy but that is also difficult to optimize tuning hyperparameters directly related to the dataset, such as balancing class weights or changing the classification threshold. On the other hand, the Random Forest model shows, again, outstanding results on the reboiler, condensate tank and generator but disappointing ones on the M014 and M015 pump. The scores regarding the M014 pump shows that the model had some skillfulness, so further improvement can be made on them, an scenario that is not applicable to the M015 pump, where the model presents no skill as shown by its F1, AUC and AP scores.

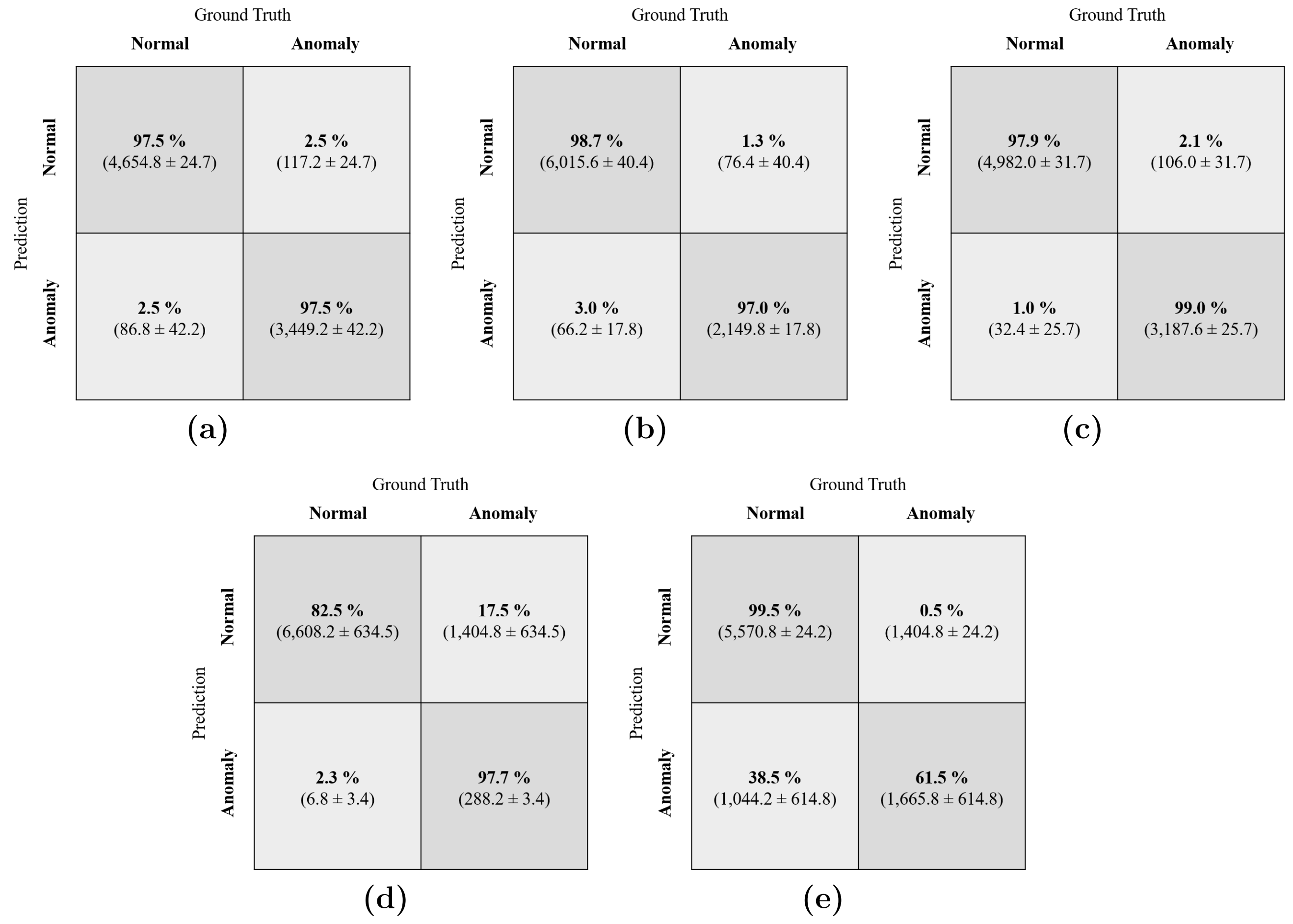


Figure 6.6: CLM confusion matrices for **(a)** Reboiler (node 0) **(b)** Condensate tank (node 1) **(c)** M014 pump (node 2) **(d)** Generator (node 3) **(e)** M015 pump (node 4).

		proposed CLM			**MLP**			**Random Forest**		
		F1	AUC	AP	F1	AUC	AP	F1	AUC	AP
Reboiler (node 0)	Mean	**0.969**	**0.996**	**0.996**	0.637	0.741	0.484	1.000	1.000	1.000
	std	0.011	0.002	0.002	0.124	0.091	0.181	0.000	0.000	0.000
Condensate tank (node 1)	Mean	**0.970**	**0.994**	**0.993**	0.436	0.639	0.279	1.000	1.000	1.000
	std	0.012	0.005	0.005	0.000	0.000	0.000	0.000	0.000	0.000
M014 pump (node 2)	Mean	**0.973**	**0.997**	**0.997**	0.549	0.689	0.378	0.664	0.788	0.784
	std	0.020	0.002	0.002	0.000	0.000	0.000	0.003	0.001	0.001
Generator (node 3)	Mean	**0.365**	**0.961**	**0.961**	0.065	0.517	0.034	1.000	1.000	1.000
	std	0.132	0.012	0.010	0.000	0.000	0.000	0.000	0.000	0.000
M015 pump (node 4)	Mean	**0.673**	**0.984**	**0.985**	0.370	0.577	0.545	0.423	0.452	0.419
	std	0.197	0.008	0.008	0.013	0.003	0.003	0.003	0.003	0.003

Table 6.4: Comparison of the proposed SLM model, MLP and RF precision-recall scores for component-level health assessment. In bold: proposed model results. Underlined: best performance values

6.3 System components relationships module results

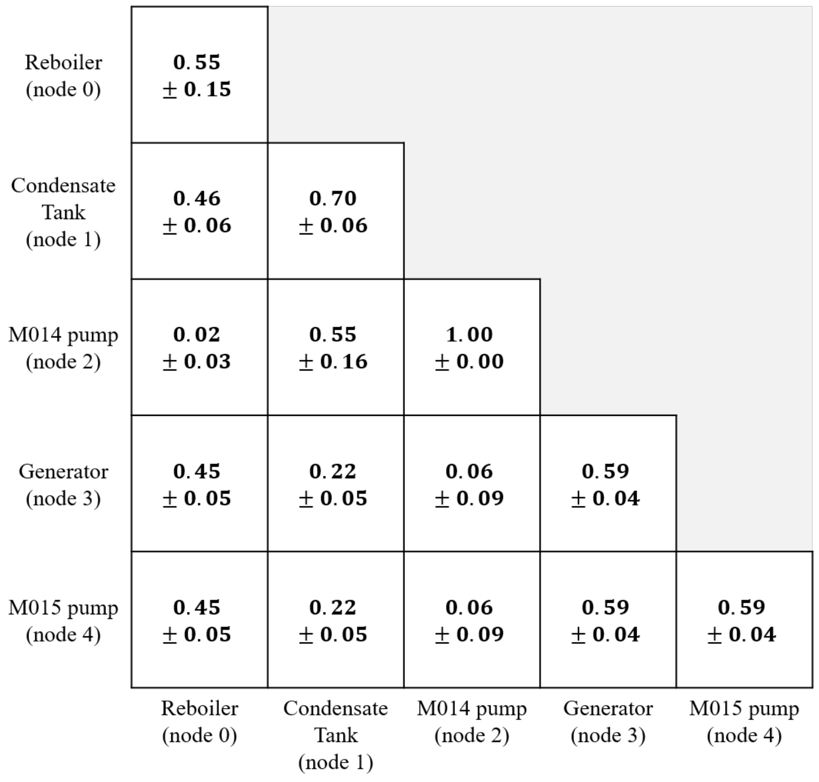
The Weighted Relationship Matrix computed from the SCRM architecture is presented in figure 6.7. The author propose this result as a way of exploding Graph Neural Networks capabilities on system-level PHM, where not only a direct health diagnosis can be made in the system and component level (as shown previously in this chapter), but also enabling an analysis of the relationships between components when the system present an abnormal behavior. As explained in Chapter 4, the WRM is trained using sensor measurements (in the form of a feature matrix X) and physical components relationships (in the form of an adjacency matrix A) regarding only anomalous data. Given this, the WRM is trained to re-construct the graph that represents the system adjacency matrix in anomalous behavior scenarios but in a way that is also dependant on the actually available sensors and its measurements. Then, explicit existence of connections represented by 0s and 1s in the original adjacency matrix are now weighted with respect of the sensor measurements, showing two phenomenons:

- Previously explicit connections (represented as 1s in A) can now diminish in value, showing that, even though a real-life physical connection is present between two sets of two components, the relationship between one set can be less relevant regarding an anomalous behavior of the system with respect to the other set.
- Using the same reasoning, previously non-existent physical connections between two components (represented as 0s in A) can augment in value, showing that both are related in operational terms when the system presents an anomalous behavior.

It is important to notice that the WRM is scaled, so every element in it is comparable in terms of value. Also, its expected that real-life physical connections between components presents higher values in the WRM that non-existent ones.

Analyzing the figure 6.7, it can be seen that the M014 pump is the most relevant component in terms of monitoring when an anomalous behavior is present on the system. The connection of the M014 pump with the condensate tank give information (given the value of 0.55 in the WRM), but its high standard deviation shows that this is not that relevant. On the other hand, monitoring directly the M014 pump is recommended to perform a better health assessment of the system. Herewith, the Condensate tank by its own also shows to be a relevant component for the system health assessment, so the same recommendation is made. On the other hand, the sub-set reboiler, generator and M015 pump (present as a triangle graph as it can be seen in figure 5.4 and that will be referred as *RGP set*) do not show to have high relevance compared to the previously mentioned components given the actually available sensors used to monitor them. Given this, and the analysis presented in the previous section regarding the generator and the M015 pump, it is recommended to monitor the pipelines that connects the *RGP set*.

Finally, regarding non-existent connections that augmented in value in a relevant way, it is of high interest to study the connections between the condensate tank with the generator and the M015 pump, which both presents a weight value of 0.22. Even though no physical connections are present between them, these three components are connected to the reboiler, by pipelines that move fluid that comes out of it. The health assessment of the condensate tank is well addressed by the framework as showed in the previous section, so, knowing this and that there is no sensor measuring of the pipelines between the reboiler and both the generator and the M015 pump, the same type of sensor is recommended to use to monitor them, this is, a temperature sensor (as the TI8015 in figure 5.2) of the output fluid of the reboiler that goes to both components.



	Reboiler (node 0)	Condensate Tank (node 1)	M014 pump (node 2)	Generator (node 3)	M015 pump (node 4)
Reboiler (node 0)	**0.55 ± 0.15**				
Condensate Tank (node 1)	**0.46 ± 0.06**	**0.70 ± 0.06**			
M014 pump (node 2)	**0.02 ± 0.03**	**0.55 ± 0.16**	**1.00 ± 0.00**		
Generator (node 3)	**0.45 ± 0.05**	**0.22 ± 0.05**	**0.06 ± 0.09**	**0.59 ± 0.04**	
M015 pump (node 4)	**0.45 ± 0.05**	**0.22 ± 0.05**	**0.06 ± 0.09**	**0.59 ± 0.04**	**0.59 ± 0.04**

Figure 6.7: Weighted relationships matrix.

Chapter 7

Concluding Remarks

Crucial industries have collected massive amounts of operational data over the last years. Nowadays, deep learning techniques have taken advantage of this to develop new health-assessment frameworks in the context of prognostics and health management, focusing mainly on components and equipment. To expand the capabilities of PHM to system-level health assessment, new deep learning techniques must be explored, being the recent development of geometrical deep learning, and in particular, graph neural networks, a promising tool to tackle this. The use of graphs allows the developing of deep learning models that can take into account not only the sensor measurements of different locations and/or components of a system, but also the different relationships (such as physical or operational) between these: one of the main problems when analyzing the health of a whole system.

To tackle the aforementioned challenge, a graph convolutional neural network based framework is proposed in this work for system-level PHM, delivering information of the health state of a system in three different levels: system-level health diagnosis, component-level health diagnosis and quantification of the mutual impact among components when the system behave abnormally. Although managing a real-world system dataset, as the chlorine dioxide generation reboiler system used in this work to validate the proposed framework, can be done using a graph structure as showed in this thesis. Furthermore, this dataset was not on the best conditions for immediate use, needing a hard-work preprocessing to label the health state of not only the system, but also the different components analyzed in it. With this, the following conclusions about the proposed framework can be made for this work.

First, regarding the system-level module, outstanding results were obtained. In terms of accuracy, the obtained confusion matrix shows not only high rates of identification of normal and anomalous behaviors of the system, but also shows to outperform other models such as a MLP and Random Forest. Furthermore, the use of graph convolutional layers for feature extraction from the input graph highly improved the performance of an MLP without it. Concerning $P-R$ scores, the proposed architecture for the SLM shows to be a highly skillful model for system-level health diagnosis in its own and compared with other techniques. Nevertheless, a main limitation when this module was studied was the use of a health-state label based on the reboiler as indicator of an abnormal behavior of the system. Ideally, for future work in this dataset (or as guideline for system-level diagnosis), system-level labels must be developed based on metrics regarding the whole system, such as output flows or efficiency of production, among others.

Second, regarding the component-level module, the proposed architecture shows high accuracy and $P - R$ scores by itself an compared to other techniques like MLP and Random Forest. A first benefit is that this module, when trained, delivers an output for the heath state fo each component, which is not the case of an MLP or Random Forest models, which needs an independent architecture trained for each system component. Even though the obtained results where, overall, of high accuracy, the Generator and M015 pump had misdirection's in the true positive (normal behavior) and negative (anomalous behavior) rates respectively. It was shown that this could be explained by the CDFs of the sensors used to measure their operation, where not much information variability was visible compared to other components that were well addressed by the module, such as the reboiler. Given this, it is recommended to monitor these equipment with different sensors that will also improve the diagnostic capabilities of the module when the semi-supervised component health state labeling procedure is done.

Third, regarding the relationships between the different components of the system, it can be seen that the use of the proposed weighted relationships matrix can give very interesting insights on the problem. It can be seen that both the M014 pump and condensate tank are the components that have more impact to the system when an anomalous behavior is observed, so they should be closely monitored. Furthermore, another interesting result concerns the impact that appears between the condensate tank and both the generator and M015 pump, which are not physically connected but share the same intermediate equipment: the reboiler, so, given the results obtained in the CLM, a sensor is recommended to be used on the pipeline between the reboiler and both the M015 pump and generator. Lastly, it is important to notice that an undirected graph was used, so no causality is present on the WRM, which must be added for a future work on the subject.

Finally, the proposed graph neural network based framework seems as a very promising tool to tackle system-level PHM on real-world systems as shown in this work. As shown above, some limitations are found and need to be addressed, so, for future works, the use of undirected graph methodologies and better labeling techniques will be done.

Bibliography

[1] Roy Assaf, Phuc Do, Philip Scarf, and Samia Nefti-Meziani. Diagnosis for systems with multi-component wear interactions. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 96–102. IEEE, 2017.

[2] Eric Bechhoefer. A quick introduction to bearing envelope analysis, 2016.

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

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[6] ZhiQiang Chen, Chuan Li, and René-Vinicio Sanchez. Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, 2015, 2015.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[8] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

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

[10] Xiaojie Guo, Liang Chen, and Changqing Shen. Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. *Measurement*, 93:490–502, 2016.

[11] Te Han, Chao Liu, Linjiang Wu, Soumik Sarkar, and Dongxiang Jiang. An adaptive spatiotemporal feature learning approach for fault diagnosis in complex systems. *Mechanical Systems and Signal Processing*, 117:170–187, 2019.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] F.O. Heimes. Recurrent Neural Networks for Remaining Useful Life Estimation. *Prognostics and Health Management, 2008. PHM 2008. International Conference on*, 2008.

[14] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

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

[16] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. sep 2016.

[17] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[18] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[19] Daeil Kwon, Melinda R Hodkiewicz, Jiajie Fan, Tadahiro Shibutani, and Michael G Pecht. Iot-based prognostics and systems health management for industrial applications. *IEEE Access*, 4:3659–3670, 2016.

[20] Y LeCun and Y Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

[21] Xiang Li, Qian Ding, and Jian Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 2018.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Kenneth A Loparo and KA Loparo. Bearing data center. case western reserve university, 2013.

[24] Ceena Modarres, Nicolas Astorga, Enrique Lopez Droguett, and Viviana Meruane. Convolutional neural networks for automated damage recognition and damage type identification. *Structural Control and Health Monitoring*, page e2230, jul 2018.

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

[26] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[27] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*,

10(3):e0118432, 2015.

[28] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[29] Weixiang Sun, Jin Chen, and Jiaqing Li. Decision tree and PCA-based fault diagnosis of rotating machinery. *Mechanical Systems and Signal Processing*, 2007.

[30] David Verstraete, Andrés Ferrada, Enrique López Droguett, Viviana Meruane, and Mohammad Modarres. Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. *Shock and Vibration*, 2017.

[31] Jinjiang Wang, Junfei Zhuang, Lixiang Duan, and Weidong Cheng. A multi-scale convolution neural network for featureless fault diagnosis. In *2016 International Symposium on Flexible Automation (ISFA)*, pages 65–70. IEEE, 2016.

[32] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[33] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[34] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[35] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[36] Dmitry Zinoviev. *Complex Network Analysis in Python: Recognize-Construct-Visualize-Analyze-Interpret.* Pragmatic Bookshelf, 2018.