4-25-2019 2:00 PM

CNN-PDM: A Convolutional Neural Network Framework For Assets Predictive Maintenance

Willamos Silva, *The University of Western Ontario*

Supervisor: Capretz, Miriam, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

Predictive Maintenance (PdM) performs maintenance based on the asset's health status indicators. Sensors can measure an unusual pattern of these indicators, such as an increased motor's vibration level or higher energy consumption, and, in most cases, failures are preceded by an unusual pattern of these measurements. The increasing number of sensors are generating a massive amount of data. Machine Learning (ML) techniques can capture and learn patterns from the data and create models to evaluate the required health indicators. These health indicators are obtained by sensors, which are becoming more present every day. Convolutional Neural Network (CNN) is a Machine Learning technique capable of extracting data representation. This ability means that less feature engineering needs to be done when compared with conventional ML techniques. This thesis presents a CNN framework to tackle assets predictive maintenance problem and a method to transform 1-dimensional (1-D) data into an image-like representation (2-D). A data transformation step is very important to make the use of CNN feasible. To evaluate the proposed framework two datasets were obtained from fans, with distinct electrical pattern, from the CMLP building at Western University. The data was preprocessed, transformed in a image-like representation and fed to a tuned classifier. The results presented by the CNN-PdM framework showed that the combination of CNN with the proposed data transformation method outperformed traditional machine learning techniques (Random Forest, Support Vector Machine, and Multi-Layer Perceptron). Statistical tests were performed to ensure that the results are statistically different from each other. The model created by the CNN-PdM framework achieved accuracy rates as high as 98% for one of the datasets and 95% for the other.

Keywords: Predictive Maintenance, Machine Learning, Deep Learning, Convolutional Neural Networks, CNN, Data Transformation.

Acknowlegements

I want to thank my “brother” Jamisson Freitas and Dr. Miriam Capretz, whose presented the chance for me to join this Master’s program. Jamisson introduced me to Dr. Capretz, and without both of them, nothing would be possible. Dr. Capretz provided the opportunity to study under her and kept pushing me forward all the time during these last two years.

I’m grateful for the numerous talks that I had with Dr. Katarina Grolinger as well. She helped me a lot in understanding machine learning, and our discussions improved my knowledge about deep learning.

I would also like to thank my colleagues and friends that helped me during the research and the life in Canada as well: Jose and Karla, Wilson and Ana, Roberto and Andrea, Wander and Leticia, Norm, and Santiago. I’d also like to thank the postdocs that helped me during these past years: Hany, Mahmoud, and Ahmed.

Achieving this without friends from Recife would be impossible. Their support kept me always moving forward, despite how hard it could sometimes be. Andre, Dayvson, Doron, Guga, and Renan, thank you for the good vibes that you kept sending me in the past couple of years. You are my brothers!

I would also like to show special gratitude to my family. My mother, Zeze, my brother, Thiago, and my wife, Thamiris. Thank you for your support along the way. These two years would be much harder without you by my side. My cousin, Rafael and his wife, Myllena: your support was crucial to me. Rafa, you are my brother! To all my aunts, uncles, and cousins: thank you very much!

Contents

List of Figures

List of Tables

List of Acronyms

ML	Machine Learning
API	Application Programming Interface
1-D	1 dimension
2-D	2 dimensions
IoT	Internet of Things
CNN	Convolutional Neural Networks
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
RF	Random Forest
MLP	Multi-Layer Perceptron
conv	convolutional
max	maximum
min	minimum
sd	standard deviation
GAF	Gramian Angular Field
CNN_MULT	CNN experiment using multiplication as feature transformation method
CNN_GAF	CNN experiment using GAF as feature transformation method
RF_NFS	Random Forest experiment without considering feature selection
RF_FS	Random Forest experiment considering feature selection
SVM_NFS	SVM experiment without considering feature selection
SVM_FS	SVM experiment considering feature selection
MLP_NFS	MLP experiment without considering feature selection
MLP_FS	MLP experiment considering feature selection

Chapter 1

Introduction

1.1 Motivation

Nowadays, asset maintenance plays a vital role in companies and countries [1], because, if an asset breaks, it can cause long downtimes that may affect companies' production costs or the comfort level of building occupants. Asset management consumes a large portion of the budget established by countries and companies. In Canada, the Federation of Canadian Municipalities (FCM) has created a five-year, $50 Million program to support municipalities to make decisions on infrastructure investment based on sound asset management practices [2]; Europe reached a record high of EUR 25.2 trillion spent in asset management in 2017 [3]. Asset management is more than just asset maintenance, it considers strategy, safety, environment, and human factors as well [4]; however, maintenance and terotechnology (ability to maintain the assets operating optimally) are among the most important competencies for assets management, having a substantial impact in a cost perspective [5]. Consequently, a solution that can improve the handling of asset maintenance can have a considerable impact on the budget of companies and governments.

There are three main approaches to address the issue of asset maintenance: corrective (also known as run-to-failure, henceforth R2F) [6]; preventive (or time-based) [7]; and predictive

maintenance (PdM or status based) [8]. In R2F, maintenance only takes place after a failure occurs. A critical asset failure, such as a broken heating pump in the winter, can lead to pipes bursting, and may cause severe damage to a building. Preventive maintenance works under a pre-existing maintenance schedule usually provided by the asset's manufacturer [7, 9]. The drawback of this approach is that the asset may not present any problem but the maintenance will be made regardless, resulting in "unnecessary" costs. On the other hand, PdM performs maintenance based on the asset's health status indicators. Sensors can measure an unusual pattern of these indicators, such as an increased motor's vibration level or higher energy consumption, and, in most cases, failures are preceded by an unusual pattern of these measurements [10]. This thesis will investigate the use of PdM in buildings.

In order to apply PdM, two approaches are commonly used: a mathematical model [11] and machine learning (ML) techniques [8, 10]. A model is a representation of a given phenomenon or system, which can be made using a set of equations (mathematical) or an ML technique. A mathematical model usually requires a domain specialist that understands the asset and its environment. The creation of a mathematical model for an asset is a very specific task; its main drawback is that the same model cannot be used for different assets. Two distinct mathematical models had to be created by Bujak [12] and Zhang et al. [13], for example, in order to represent two different boilers. ML techniques can capture and learn patterns from the data and create models to evaluate the required health indicators. As buildings have numerous pieces of assets that must be monitored, the usage of ML is more suitable because it enables the creation of a specific model for each one of the assets using the same ML techniques.

An important aspect of modeling is to work with the available features. Features are the different data inputs (individual measurable properties), such as temperature or energy consumption [14]. Dealing with these inputs may be a challenge by itself if two or more highly correlated features are available because they may introduce bias to the model [14]. The feature selection step is also significant in order to determine which of the available data inputs are relevant to the PdM; the most suitable set of features must be selected for a particular problem.

In a scenario in which two different maintenance teams work with the same asset, for instance, one team may be interested in one set of features and handle only electrical issues with the asset, while another team may be interested in mechanical features, such as vibration, and be specialized in mechanical problems.

Different ML techniques perform better under different scenarios. There are ML techniques that work better with a smaller amount of data, but are outperformed in situations in which the amount of data drastically increases [15]. Neural Networks (NNs) [16] are one class of ML techniques that are a combination of cells called neurons, which are organized in layers. The number of layers of an NN relates to the NN's *depth*. The deeper the NN is, the more layers (and also neurons) the NN will have. A particular type of NN is the Convolutional Neural Network (CNN) [15]. The CNNs try to mimic the human brain ability to classify objects by their shape. For this reason, CNNs are most commonly used for image classification problems. In other domains, data can be represented as an image through a feature transformation process. In this research, data transformation methods are used to represent data as images.

1.2 Contribution

This work proposes the Convolutional Neural Network Predictive Maintenance (CNN-PdM) framework. The CNN-PdM aims at tackling the PdM problem using CNN as its core. The usage of CNN is a crucial factor because this technique is capable of creating data representation from the data's features, which is a significant advantage when compared to traditional machine learning techniques as it removes the need to perform feature engineering. Since PdM is an approach that have the assets' health status constantly monitored, a drift from normal behavior indicates that the asset needs maintenance. This study shows that CNN can better identify abnormal patterns from the data and provide more accurate advice about the need to perform PdM in assets.

Nowadays, there is a collection of sensors in assets that can offer a vast amount of data,

from which CNN can benefit and outperform previous state-of-the-art ML models. More data represent the possibility to have deeper CNN models, which leads to higher level feature maps that can be created by the CNN layers; this feature map representation is the main reason why they outperform conventional ML models as the amount of data increases. A lack of research in this area suggests that there is no previous usage of Convolutional Neural Networks (CNN) with electrical data and also in the PdM domain. A data transformation method has also been proposed to represent 1-dimensional data (1-D) into 2 dimensions (2-D). The multiplication method, introduced in this research, was able to represent the data in a more heterogeneous way than the previous data transformation methods; this transformation method combined with the feature representation from CNNs outperformed conventional ML approaches.

The proposed framework in this thesis was evaluated in two case studies. These case studies compared CNN-PdM framework using two different image transformation methods and CNN with well-known ML techniques that have been applied in the PdM domain. For each case study, one dataset was used for all the experiments. Each one containing data from different assets (SF103B and RF101) from Claudette MacKay-Lassonde Pavilion (CMLP) building at Western University, London, Canada. Each dataset contains data from February 1, 2018 to January 31, 2019.

The results highlight the following observations:

- For comparison purposes, 8 experiments were conducted in this research to assess how well the CNN-PdM framework would perform when compared with traditional ML techniques used in the PdM domain (Support Vector Machine and Random Forest). Since CNN is a type of neural network (NN), we are comparing it with Multi-Layer Perceptron, which is one of the most commonly used NNs. The evaluation was performed under 30 simulations for each experiment in two case studies. For each case study, 30 simulations using the same data were performed for each experiment; the CNN combined with the data transformation technique proposed in this work outperformed the other ML approaches. In the first case study, the CNN-PdM framework outperformed

the other approaches by at least 1.91% after, while in the second one, it outperformed the other approaches by at least 1.5% using also the same data for the experiments.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 covers relevant background topics that are necessary to comprehend the remaining content of this work. First, this chapter provides information related to machine learning algorithms used in this work; second, this chapter includes a literature review on current studies that use CNN for various domains other than image classification and research that discusses PdM. Finally, it offers distinctions between this thesis and previously published scholarship.

- Chapter 3 discusses the stages of the CNN-PdM framework, which consists of four stages: *Data Collection*, *Data Preprocessing*, *Data Transformation*, and *CNN Model Creation*. This chapter presents the steps used in each of these stages.

- Chapter 4 describes the implementation performed for the CNN-PdM framework. This chapter also includes the libraries and methods used for the implementation used in this research. It also presents the experiments conducted using the CNN-PdM framework along with a comparison with other ML techniques. Two experiments were performed that considered two distinct assets: two fans from the CMLP building at Western University that are used for different purposes; therefore, these assets' electric profile (energy consumption, amps, real power, and so on) varies from one to another.

- Chapter 5 provides the outcomes of this thesis, as well as discussion regarding Predictive Maintenance and directions for possible future work.

Chapter 2

Background and Literature Review

This chapter has two main objectives: first, it focuses on providing key background information to help in the understanding of the CNN-PdM framework; second, it presents the literature review that discusses what has been done on topics related to CNN and PdM.

2.1 Background

This section presents basic concepts of machine learning techniques used in this work: Convolutional Neural Networks (CNN), Data Transformation Tecnniques, Random Forest (RF), Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP). This section also provides information related to feature transformation and Deep Learning.

2.1.1 Machine Learning

Machine learning (ML) is a field of Artificial Intelligence (AI). ML systems can extract knowledge from data [17] through one of four main learning paradigms: supervised, semi-supervised, unsupervised, and reinforcement learning [18].

In supervised learning problems, the available data contain the examples (inputs) and also the target values (outputs), while in semi-supervised problems only a portion of the data con-

tains the expected outputs. The two most common types of supervised and semi-supervised learning problems are classification and regression. In classification problems, the ML model's output is one of a finite number of classes for a given input. In image classification, one possible problem could be, for example, to detect if an object is a cat or not. In this case, it is also called a binary classification problem, because there are only two possible classes as an output. Another classic classification example could be to perform the classification of the iris flower dataset, in which the possible outputs are setosa, virginica or versicolor. This type of problem is known as a multi-class classification problem. On the other hand, in regression problems, the ML model's output is characterized by one or more continuous values. Possible examples of regression problems are to perform the prediction of energy consumption of an asset for the next day or to predict Microsoft's stock value within the next week. To evaluate classification problems, some metrics are commonly used, such as accuracy (Equation 2.1), precision (Equation 2.2), recall (Equation 2.3) and f1 score (Equation 2.4). These metrics are described below:

$$accuracy = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} (\hat{y}_i = y_i) \tag{2.1}$$

$$precision = \frac{tp}{tp + fp} \tag{2.2}$$

$$recall = \frac{tp}{tp + fn} \tag{2.3}$$

$$f1\_score = 2 * \frac{(precision * recall)}{(precision + recall)} \tag{2.4}$$

where $\hat{y}_i$ is the predicted class, y_i is the expected class, tp stands for true positives, fp for false positives, and fn for false negatives.

Unsupervised learning is characterized by the absence of labels on the training data. Clustering and density estimation are the most common unsupervised learning problems. Clustering is based on the act of grouping examples that share similarities and are usually used as a way to present insight about the data. Density estimation relates to finding the data distribution in the space. The iris flower dataset can also be handled as a clustering problem if the examples'

classes are not provided.

Finally, reinforcement learning relates to unsupervised learning in the sense that ideal outputs are not provided. The reinforcement learning focuses on finding the optimal output through a trial and error process. Instead of predicting values, the output of reinforcement learning techniques is actions. These actions are evaluated by an agent that interacts with the environment in which it is placed. The evaluation of these actions on the environment works as a feedback for the model, that can be a reward or a punishment. Volodymyr *et al.* [19] presented a reinforcement learning framework that was tested on multiple ATARI games, in which the pixels from the screen are fed to the agent which learns how to play those games.

Deep Learning is a particular type of ML that presents a unique capability of extracting data representation. This ability signifies a big step for ML data preprocessing, as a person may not perform the feature engineering step when using deep learning techniques. Some well-known deep learning techniques are Convolution Neural Networks, Recurrent Neural Networks, Deep Belief Network, and Stacked Auto-Encoders [20].

As the name suggests, Machine Learning techniques need to learn (from the data). The learning stage is called training stage. The data is usually split into two or three subsets: training, validation (optional), and testing sets. The ML model will learn from the training set, be affirmed from the validation set, and evaluated within the testing portion of the data. One could use 80% of the data to train the model and 20% to test (evaluate) it, for instance. This would be refereed as a 80/20 training/testing data split. Another example would be to have a validation step. In this case, the data could be split as 60/20/20, that is 60% of the data would be used to train, 20% to validate, and 20% to test the model.

2.1.2 Convolutional Neural Networks

Deep Learning (DL) is a subset of ML techniques. The main advantage of deep learning techniques is that there is almost no need to perform feature engineering, which is to create features by hand, usually by a domain expert [21]. This advantage emerges from the capability of per-

forming representation learning, which means extracting knowledge from raw data. The most commonly used DL technique for image classification is the Convolutional Neural Network (CNN).

Inspired by animals' visual cortex, they have the capability of extracting strong spatially local correlation to create high-order features information from the raw data [22]. It is possible to arrange CNN neurons in a three-dimensional structure to capture information regarding width, height, and depth; therefore, CNNs are commonly used to perform image classification.

CNNs usually present a structured pattern of layers, combining successive convolutional (conv) layers with ReLU (rectified linear unity as activation function, sometimes seen as a layer by itself) and pooling layers, represented as a box in the hidden layers stage in Figure 2.1.

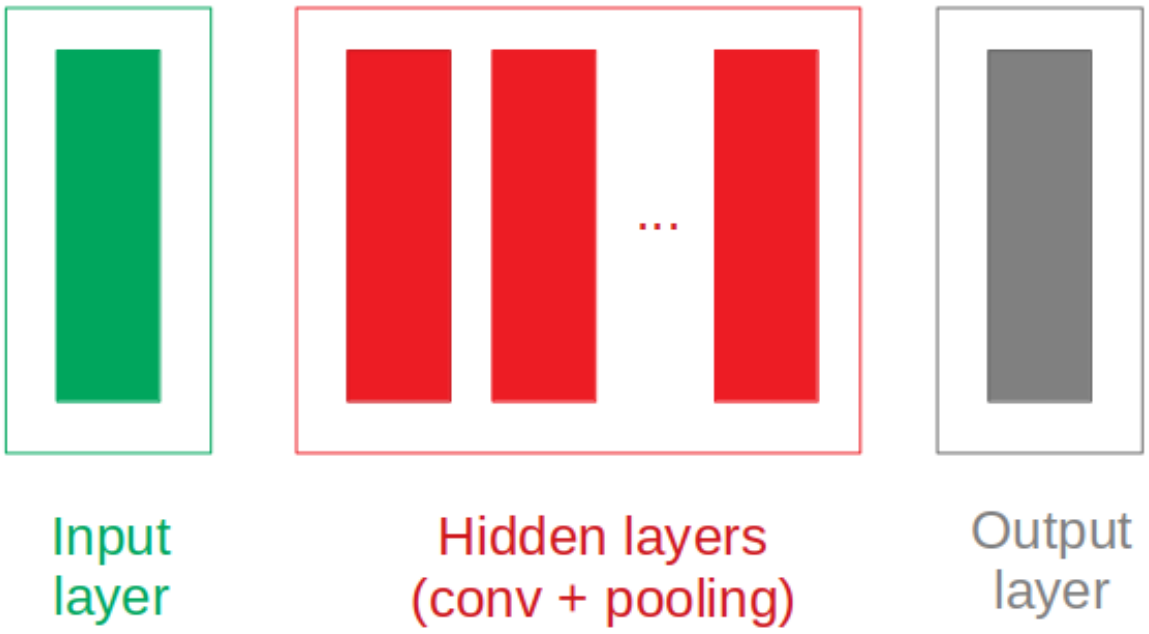

Figure 2.1: Commonly used CNN architecture: input layer, combinations of conv and pooling (hidden) layers and output layer.

The convolution operation performed by the convolutional layer is the feature extractor, which is a filter (also known as a kernel) that slides over data, combining information. Each subsequent layer increases the abstraction level of the features captured by the filters. The

filters in the first layer, for instance, would be responsible for detecting the presence of edges, while the second layer would be in charge of detecting the combination of edges. The third convolutional layer should be able to detect parts of familiar objects. Note that the abstraction level increases from layer to layer [21].

The sliding step distance of the filter is known as stride [23]. Usually, the convolution operation uses stride equals to one, which means that the kernel was shifted by one position. The output of this filter is the dot product from the filter elements and the data within the kernel boundaries. This operation can be interpreted as a forced merging of similar features into one single feature [21].

To avoid overfitting, a very effective technique is used: dropout. A dropout layer is responsible for randomly changing the output of a fraction of neurons to zero, a fraction is known as dropout rate. An input array $[0.1, 0.5, 0.7, 0.3, 0.4]$ submitted to a dropout rate of 0.2 (20%), for example, would present one of its value set to zero. Dropout is performed only during the training stage. During testing, the output values are scaled down by a factor equal to the dropout rate in order to balance the fact that more units are being used than during the training period.

2.1.3 Data Transformation

As it was previously presented, CNN was created and outperforms conventional machine learning techniques in image classification problems. Data transformation is a step performed usually to present data in an easier way to be visualized. A common use of data transformation is to perform dimensionality reduction, in which large volumes of high-dimensional data are reduced to 2 or 3-dimensional (2-D and 3-D, respectively) to provide a better understanding of the data [24, 25]; however, sometimes one needs to perform the opposite task of dimensionality reduction: to increase the number of dimensions. Mapping the 1-dimensional (1-D) data into 2-D, or even 3-D, is an important step if one considers using a CNN model. Wang and Oates [26] proposed Grammian Angular Field (GAF) to transform time-series into images aiming on

improving classification and data imputation.

- Grammian Angular Field (GAF): Given the input example $d = \{f_1, f_2, ..., f_{49}\}$, the GAF data transformation output can be obtained by applying Equation 2.5:

$$GAF = \begin{bmatrix} < f_1, f_1 > & \cdots & < f_1, f_{49} > \\ \vdots & \ddots & \vdots \\ < f_{49}, f_1 > & \cdots & < f_{49}, f_{49} > \end{bmatrix} \tag{2.5}$$

  where $< f_i, f_j >$ stands for $f_i \cdot f_j - \sqrt[2]{1 - f_i^2} \cdot \sqrt[2]{1 - f_j^2}$. An example of GAF features' transformation can be seen in Figure 2.2.

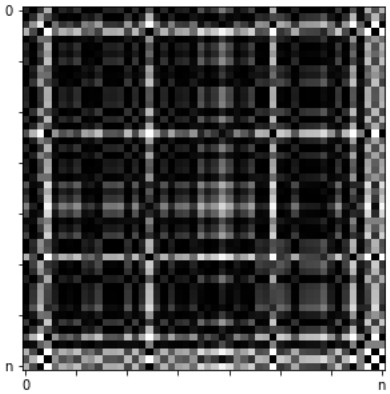

Figure 2.2: An example of sensor data represented as an image (GAF).

Chen *et al.* [27] proposed other two techniques to perform data transformation: Moving Average Mapping (MAM) and Double Moving Average Mapping (DMAM). For a comparison purpose, the authors have compared the proposed methods with GAF.

- Moving Average Mapping (MAM):

  Moving Average is a well-known technical analysis indicator in the financial domain. The purpose of moving average is to create a smoother curve for the time-series; this

increases the temporal dependency. The authors translated this idea into a mapping algorithm. Their primary goal was to capture multiple moving average values considering different time frames for each input, as illustrated in Equation 2.6:

$$MAM = \begin{bmatrix} MA_{t-D+1,1} & \cdots & MA_{t,1} \\ \vdots & \ddots & \vdots \\ MA_{t-D+1,D} & \cdots & MA_{t,D} \end{bmatrix} \tag{2.6}$$

where $MA_{i,j}$ is the average stock closing price from period i to j.

- Double Moving Average Mapping (DMAM):

  The authors wanted to benefit from more information than only one feature (closing price) over time. Instead of using the closing price, the authors utilized the average value from opening and closing prices were used. The aim was that the average value from opening to closing aggregated more information than only the last one.

MAM and DMAM apply the concept of Moving Average, which represents the variation over time of a single feature (in DMAM's case, the average value of two features). Those techniques are not applicable to the PdM domain when considering different features, such as energy consumption, amps, or a motor's vibration level. GAF seems to be the most appropriate technique to use in the PdM domain. Instead of capturing the correlation among a single feature over time, it could be used to capture the relationship among features from a specific time; it is possible to observe, however, that the image representation presented by GAF is dark and might be a problem, as it presents low contrast among the transformed features (the subtraction term $\sqrt[2]{1-f_i^2} \cdot \sqrt[2]{1-f_j^2}$ decreases the pixel intensity). In order to prevent it, a multiplication features' transformation has been proposed in this research.

2.1.4 Random Forest

Proposed by Breiman [28], Random Forest (RF) is a technique which creates a set of N decision trees $\{T_1(F), T_2(F), ..., T_N(F)\}$, where F is a feature dimension vector of size m given by $F = \{f_1, f_2, ..., f_m\}$. Each tree is trained by different subsets of the dataset. Usually, decision trees are created by splitting each node with the best possible variable, while in RF the node is split using a subset of features randomly generated for that specific node [29]. This extra randomness makes RF less prone to overfitting than decision trees. In a classification problem, the output class is given by the majority vote of the trees in the ensemble, while in a prediction problem the output is the average value of each tree prediction [30].

2.1.5 Support Vector Machine

Support Vector Machine (SVM) is a commonly used approach in binary classification problems [8]. Given a training set denoted by $S = \{x_i, y_i\}_{i=1}^{n}$, SVM is responsible for mapping the input data into a higher dimensional space (greater than n), where it is easier to separate the examples with a hyperplane [31]; the examples that are closest to the hyperplane are called support vectors. The hyperplane is surrounded by a margin, which represents how fat this hyperplane can be without misclassifying the examples; the larger the margin, the better it is, because new examples can be within the margin without being misclassified.

2.1.6 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is a type of feed-forward Neural Network. Its main component, called neuron, is grouped in layers. Each neuron can receive multiple inputs from the previous layer, and each input has a weight. The sum of the weighed inputs is modified by a non-linear function, called activation function. The output of the activation function is used as an input by the neurons in the subsequent layer. The non-linearity of the activation function allows the MLP to approximate non-linear functions [32]. MLP presents at least three layers: input,

hidden and output. MLPs may present more than one hidden layer [22].

2.2 Literature Review

This section provides a literature review and is divided in three parts. The first covers works in which CNN is used to perform different tasks other than image classification. The second present works that use mathematical models in the PdM domain, while the last one offers researches that investigate the creation of ML models in PdM domain.

2.2.1 CNN in Different Domains

CNN was built mostly to perform image classification; however, it has been applied to other domains [33, 34, 27]. Seo and Cho [33] used CNN to perform offensive sentence classification as this domain is susceptible to data imbalance. The authors benefited from the transfer learning capability of CNN and perform its training using an artificial dataset to avoid imbalance issues. They showed that even though CNN does not perform well in offensive classification as datasets are usually not enough to train the features (CNN filters), training on an artificial dataset makes it possible to overcome this issue.

Muckenhirn *et al.* [34] presented a classification model to deal with voice presentation attack detection. The authors used a CNN as a feature extractor followed by an MLP with only one hidden layer that works as a classifier. They concluded that their research is complementary to a spectral statistics-based approach and its main advantage is that their proposed method does not use prior assumptions related to speech signals.

Chen *et al.* [27] proposed a CNN based method to analyze financial time-series. In their work, they had over one million samples of data from January 2, 2001, to April 24, 2015, and the sampling period was one minute. The objective of their research was to create a method that could detect the trend of the time series. The authors used a time-window of size 20 and compared 3 data transformation methods to represent the 1-D data into 2-D: Gramian Angu-

lar Field (GAF), Moving Average Mapping (MAM), and Double Moving Average Mapping (DMAM). The best results were achieved by the combination of GAF and CNN, with an average of 54.86% accuracy.

2.2.2 Predictive Maintenance using Mathematical Models

Authors have used mathematical models to deal with PdM [11, 35, 36]. Borgi *et al.* [11] proposed a mathematical model to deal with predictive maintenance for industrial robots. This method relied on the data analytics of the robot power time-series to predict the robots manipulator accuracy error. The data used in this work were a set of 21 features (7 for each of the 3-phase current) and 155 examples. Their work showed that a set of electrical features presented a high correlation with 3-phase current and the robots accuracy values.

Cauchi *et al.* [35] presented a mathematical model to describe the user discomfort as a consequence of degraded temperature control, which advised for maintenance of a boiler. The authors proposed two models: linear and exponential. The following features were considered as input for both models: building inside and outside temperatures, boiler efficiency, the ideal range of comfort temperature, the thermal resistance of the zone envelope, heat gained from solar radiation, and the heat gained due to occupants. Their methodology aimed to advise for the boilers maintenance or their replacement.

Olde Keizer *et al.* [36] offered a mathematical model to tackle problems of systems with multiple components. The authors demonstrated that the simple fact of duplicating policies from a single-component into multi-components systems might not be useful. Their methodology was based on a Markov Decision Process with a focus on minimizing costs. The model took into consideration state space, action space, transition probabilities, and the expected costs. The state space represented the state of each component, status of spare orders and the number of spare components. The action space represented the decisions made about components replacement. The transition probabilities would take into consideration if any component was replaced to calculate the transition to a new state. Expected costs were calculated by the

sum of operating, replacement, order, and holding costs. The results showed that ordering decisions had a higher impact on the costs than replacement decisions.

The difference between the studies above [11, 35, 36] and the proposed research is that the CNN-PdM framework uses an ML algorithm; therefore, it is simpler to accommodate different assets and even asset types than if using a mathematical model. Different ML models can be trained using the same steps and the same ML technique to work with PdM for a boiler or an HVAC, for example, while the same mathematical model would not be able to represent both assets.

2.2.3 Predictive Maintenance using Machine Learning

In contrast with the usage of mathematical models in PdM, machine learning models are also used [8, 37, 38, 39, 40, 41, 42, 43]. Susto *et al.* [8] presented an ensemble approach to detect when tungsten filaments must be replaced during ion implantation, one of the steps in the semiconductor manufacturing fabrication. The authors had access to 33 maintenance R2F cycles approach, recording 3671 examples of both normal and faulty data, containing 31 features each (9 for current, 3 for pressure, 7 related with voltage and so on). Susto *et al.* tested SVMs ensembles (PdM-SVM) and KNN ensembles (PdM-KNN); the PdM-SVM slightly outperformed the other approach.

Coraddu *et al.* [37] proposed a study on ML models that performed PdM on gas turbines (GT). The authors focused on Regularized Least Square (RLS) and SVM to predict the ideal maintenance time of the GT based on three variables: speed, compressor decay, and GT decay. The experiments were performed using an artificial dataset created by a simulator. The results showed that the SVM outperformed the RLS model.

Ma *et al.* [38] presented a Discriminative Deep Belief Network (DDBN) with its parameters optimized using Ant Colony Optimization (ACO) algorithm to deal with predictive maintenance on a gas turbine (GT). The authors used 100 different R2F datasets, containing 14 independent variables regarding the assets' health condition. The assets were classified ac-

cording to their health status into 5 categories, based on how many cycles the GT would work before it failed: $0\%-20\%$, $20\%-40\%$, $40\%-60\%$, $60\%-80\%$, and $80\%-100\%$. In this study, the ACO-DDBN model was compared with an ACO-SVM (SVM optimized by ACO) model, and the ACO-DDBN presented the best results achieving 97% accuracy on the classification over the 5 classes.

Leahy *et al.* [39] proposed an SVM approach to perform PdM on wind turbine. The authors' dataset was created from SCADA data and presented 45000 datapoints. Leahy *et al.* tried to perform 6 different fault classification: fault/no-fault, feeding faults, aircooling faults, excitation faults, generator heating faults, and mains failure. The authors used SVM hyperparameter optimization by randomized search. They achieved better results on detecting generator heating faults, classifying correctly 100% of the cases, but they obtained only 90% recall and 8% precision on the fault/no fault dataset.

A predictive maintenance for photovoltaic systems using Multi-Layer Perceptron (MLP) neural network and Triangular Moving Average (TMA) was proposed by De Benedetti *et al.* [40]. Their work is based on the prediction of the power generated by the photovoltaic system, considering the produced power as a function $f(I,T)$, where I is the solar irradiance and T is the environment temperature.

Baptista *et al.* [41] presented a framework that uses data-driven techniques and Auto-regressive Moving Average (ARMA) to deal with the predictive maintenance for a valve for an aircraft engine. The feature extraction module uses ARMA to provide a forecast of the next event. This prediction is used as input for a predictive model in a further step. 13 statistical features that are created from historical data and are combined with the ARMA's forecast as a new set of features. PCA transforms these 14 features, and the output of this transformation is fed to train the data-driven model. The authors tested f-nearest neighbors, RF, NN, SVMs, and also linear regression for the data-driven model. The SVM outperformed the other models.

Kulkarni *et al.* [42] proposed an ML base approach to perform early detection of issues on refrigeration and cold storage systems. The authors have applied a feature extraction step which

consisted of seasonality decomposition and pattern learning by using dynamic time wrapping and clustering. After that, an RF classifier was used to detect if the pattern was abnormal or not. Their method was able to achieve 89% precision.

Paolanti *et al.* [43] presented an ML model to perform PdM on a cutting machine. The authors used an RF to perform a classification among possible rotor status of the cutting machine. They had 15 available features and 530731 examples. The authors' RF model was able to achieve 95% of overall accuracy.

In contrast with the studies described above [8, 37, 38, 39, 40, 41, 42, 43], the proposed research makes use of CNN, a model that is capable of extracting feature representation to outperform classic ML techniques. With a higher number of features, finding an optimal subset of features can be hard. Only Susto *et al.* [8] presented a high number of features (31). All other approaches presented less than 14 features. CNNs are capable of creating relevant feature representation with a high level of abstraction to perform the classification and in Internet of Things' (IoT) era, where more and more data become available every day, being able to deal with massive amount of features is relevant to any ML problem.

2.3 Summary

This chapter presented an overview of ML models that are key to the understanding of the CNN-PdM framework, in particular, RF, SVM, MLP, and CNN. Works that use CNN in domains that are not related directly to image classification were also presented in this chapter. This chapter also presented numerous studies that aim at dealing with PdM by using mathematical or machine learning models.

Chapter 3

CNN-PdM Framework

This thesis proposes the Convolutional Neural Network Predictive Maintenance (CNN-PdM) framework, which is presented in Figure 3.1. The CNN-PdM framework uses historical data gathered by sensors, generates temporal features, performs data transformation, and realizes model tuning to identify possible degradation of the equipment and advice for maintenance.

The CNN-PdM framework consists of four main stages: the *Data Collection*, the *Data Preprocessing*, the *Data Transformation*, and the *CNN Model Creation*. Data transformation can be seen as a part of the data preprocessing stage, but as the Data Transformation is particular due to the CNN model, the Data Transformation is represented as a separate stage. The framework division into these stages makes the CNN-PdM easier to understand as its responsibilities are well defined among each stage. The following sections describe the stages used in this research.

3.1 Data Collection

This stage presents different sets of data that can be available from an asset. The features can be categorized as **Electrical**, such as energy consumption or current; **Mechanical**, for instance, vibration or oil level; **Contextual and Temporal**, which add information about the conditions (contextual) and time (temporal) in which the asset is working, respectively. When the asset

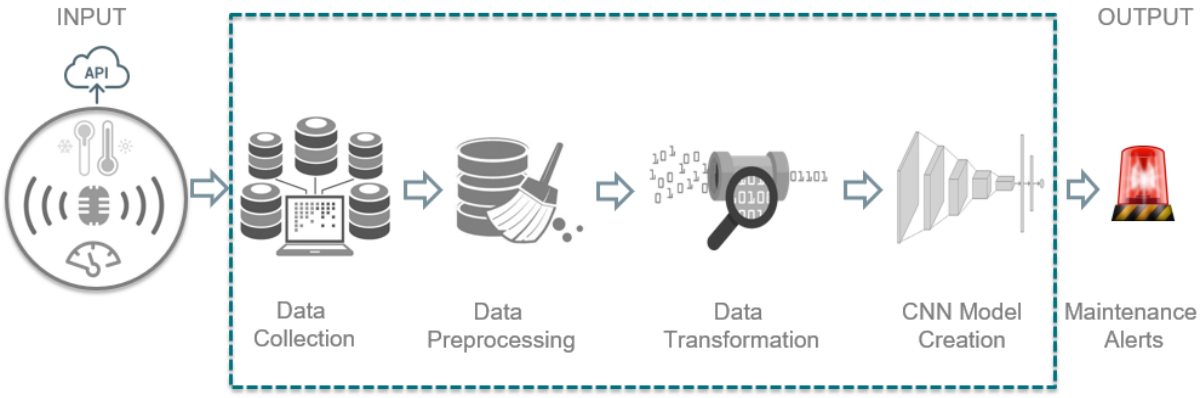

Figure 3.1: CNN-PdM framework.

performs far from what was expected under those conditions, it becomes possible to detect a problem. Outside temperature and day of the week provide information about the context in which the equipment was working in that specific time. Lower energy consumption is expected from a heating pump during the summer than during the winter, for example.

The data may be acquired by using an API (Application Programming Interface), a SCADA (Supervisory Control and Data Acquisition) system, or from any other source; sometimes multiple sources may be used, as contextual features may be collected from external sources, such as the Historical Climate Data [44], from the Government of Canada. Information such as temperature, relative humidity, wind speed, and wind direction are available in this domain.

3.2 Data Preprocessing

The preprocessing stage can be divided into multiple steps. This section presents the steps used in this work.

3.2.1 Dataset Construction

The first step of the data preprocessing is to construct the dataset with all the available data values from different features. Contextual, electrical, and mechanical features are grouped. If any other types of features are available, they should be considered in this step. In this work, a

single dataset is considered as input for the models; therefore, this step is crucial to gather all available data. The features are grouped by their timestamps to create a matrix that gathers all the data.

3.2.2 Data Imputation

The next step is to find possible missing values from the previous data matrix. Sometimes, there are missing values for some features in the data matrix. In this case, the dataset must be filled for each one of the features. It is essential to be able to fill these values in order to not discard any available example [45]. Several methods can be used in this step. Two approaches are more commonly used: to fill the missing values with average, median, or zero, or to use an interpolation technique, such as linear, time, quadratic, or cubic interpolation.

3.2.3 Feature Creation

At this step, features are created from the timestamps (date and time) information from each example of the dataset. The *Feature Creation* step adds temporal information that may be relevant to the PdM problem. In some scenarios, information regarding the day of the week or season can add value to the CNN model. The feature creation is also known as feature engineering. In 2012, Domingos [17] said that "... one of the holy grails of machine learning is to automate more and more of the feature engineering process.".

To add temporal information, Equations 3.1 and 3.2 were used to add information regarding the month, while Equations 3.3 and 3.4 provided knowledge regarding the hours of the day.

$$month\_cos = cos\left(2 \cdot \pi \cdot \frac{month}{12}\right) \tag{3.1}$$

$$month\_sin = sin\left(2 \cdot \pi \cdot \frac{month}{12}\right) \tag{3.2}$$

$$hour\_cos = cos\left(2 \cdot \pi \cdot \frac{hour}{24}\right) \tag{3.3}$$

$$hour\_sin = sin\left(2 \cdot \pi \cdot \frac{hour}{24}\right) \tag{3.4}$$

3.2.4 Normalization

The next step is to normalize the data. This step is important to bring every feature to the same range; depending on the technique used, it is possible to center the data around zero and have variance in the same order. Some techniques are sensitive to different features ranges (neural networks, for example). Performing data normalization can make the model less sensitive to bias. Normalizing the data is the last well-known preprocessing step used in this work.

The two most commonly used techniques to perform data normalization are to scale the data between an interval (MinMax scaling) or to standardize the data and center it around 0. The MinMax operation is performed accordingly to the Equation 3.5:

$$Scaled = \frac{e_i - E_{min}}{E_{max} - E_{min}} \cdot (max - min) + min \tag{3.5}$$

where e_i is the value to be scaled, E_{min} is the minimum value for the feature, E_{max} is the maximum value for the feature, *min* is the minimum scale limit (in this research it is -1), and *max* is the maximum scale limit (in this research it is 1). On the other hand, the standard scale effect is to center the feature in zero and scale it to the unit variance. Equation 3.6 presents how the standardization process is performed:

$$Scaled = \frac{(X - \mu)}{s} \tag{3.6}$$

where X is the value that is going to be scaled, μ is the mean of the samples, and s is their standard deviation.

3.3 Data Transformation

After the normalization process is completed, the *Data Transformation* stage begins. CNNs were created to handle image classification problems; therefore, it is ideal that the data examples are represented as images.

The data transformation stage is used to represent the single-dimensional (1-D) data (a feature array) as a two-dimensional (2-D) data (a feature matrix). This 2-D data can be interpreted as an image. The goal of an image-like feature representation is to capture the correlation among features of each specific example from the dataset.

Wang and Oates [26] proposed GAF, a data transformation method, which uses a single feature during an interval of time. Chen *et al.* [27] tested GAF and proposed two other methods, Moving Average Mapping and Double Moving Average Mapping, to perform feature representation in financial time-series domain. In Chen's work, a window would slide over the same feature through time. The difference between the data transformation performed in their work and what is proposed in this research is that while in their work the authors captured the correlation of one feature through time, this research considers one time step only and the correlation among different features is captured using a data transformation method proposed in this work: the multiplication method.

3.3.1 The Multiplication Method

The multiplication method was proposed to create more heterogeneous image-like data representation than the ones created when using GAF. These heterogeneous patterns in the image representation can help the creation of different filters by the CNN model, increasing its ability to correctly classify the data as normal or faulty. Given the example $d = \{f_1, f_2, ..., f_n\}$, the transformed example D_m is given by a matrix created by the multiplication of the f_i features by the whole example d, as presented by Equation 3.7:

$$D_m = \begin{bmatrix} f_1 \cdot f_1 & f_1 \cdot f_2 & \cdots & f_1 \cdot f_n \\ f_2 \cdot f_1 & f_2 \cdot f_2 & \ddots & f_2 \cdot f_n \\ \vdots & \vdots & \ddots & \vdots \\ f_n \cdot f_1 & f_n \cdot f_2 & \cdots & f_n \cdot f_n \end{bmatrix} \tag{3.7}$$

where $f_i \cdot j_j$ stands for the multiplication of f_i times f_j.

3.4 CNN Model Creation

This section presents how the CNN model should be tuned, trained and tested. The tuning step is important to find the most suitable hyperparameters for the ML problem. The training should be performed using the hyperparameters found in the previous step. Lastly, the model should be evaluated in the test step.

3.4.1 CNN Model Parameter Tuning

This step must occur to perform hyperparameter tuning for the CNN model. Different methods, such as greedy search or randomized search, can be used. In greedy search optimization, all the possible combinations of hyperparameters for the CNN model are tested, while in randomized search a randomly generated set of combinations is considered and evaluated. This step is crucial to obtain an optimized CNN model to handle the PdM problem.

3.4.2 CNN Model Training

As soon as the most suitable hyperparameters are identified, a tuned model is trained. In real applications, all the available data should be used for training the model. For research purposes, the data are split to evaluate the model performance. There are different ways to split the dataset between the training and test sets. Some common ways are 60/40, 70/30, 75/25 or 80/20 percent for training and test sets. The way to split the dataset can vary from sequential to

shuffling data. The training set should be used in this step to perform the model training, while the test set should be used to evaluate the model.

3.4.3 CNN Model Testing

The test set generated in the previous step should be used in this step. In this work, the CNN model is evaluated accordingly to four metrics described in Section 2.1.1: accuracy (Equation 2.1), precision (Equation 2.2), recall (Equation 2.3) and f1 score (Equation 2.4). The confusion matrix is also generated. Confusion matrix presents information related to the classification process. This matrix provides information regarding the number for tn, fp, fn, and tp, where:

- tn are negative examples classified correctly (as normal)
- fp are negative examples classified incorrectly (as faulty)
- fn are positive examples (faulty) classified incorrectly (as normal)
- tp are positive examples classified correctly.

In the PdM domain, the worst mistake is to classify a faulty example as normal (fn). If a datapoint is classified as abnormal, a warning must be created. This notification can be created and sent out in various ways, such as e-mail, SMS, or even in a dashboard. The a high-level algorithm of the CNN-PdM framework comprehends steps from the data collection until the new examples classification. More formally:

3.5 Summary

This chapter has introduced the CNN-PdM framework and discussed each of its core stages: *Data Collection*, *Data Preprocessing*, *Data Transformation*, and the *CNN Model Creation*. The discussion explained the responsibilities of each step and their components, and also how they interact with each other.

Algorithm 1: CNN-PdM algorithm.

Data: Hyperparameters (HP), List of Data Sources (DS)
Result: Trained Model, Metrics
// **Data Collection** line 1
1 raw_data ← collect_data(DS);
// **Data Preprocessing** lines 2-5
2 aggregated_data ← perform_data_aggregation(raw_data) ;
3 filled_data ← perform_data_imputation(aggregated_data);
4 completed_data ← perform_feature_creation(filled_data);
5 normalized_data ← perform_data_normalization(completed_data);
// **Data Transformation** line 6
6 transformed_data ← perform_data_transformation(normalized_data);
// **CNN Model Creation** lines 7-10
7 training_data, testing_data ← split_dataset(transformed_data);
8 best_hyperparameters ← perform_hyperparameter_tuning(HP, training_data);
9 trained_model ← train_model(best_hyperparameters, training_data);
10 metrics ← evaluate_model(trained_model, testing_data)
11 **return** trained model, metrics

Chapter 4

Implementation and Evaluation of the CNN-PdM Framework

This chapter presents the implementation of the CNN-PdM framework. Each section offers a stage of the framework, the libaries (libs), and methods used in this research. The implementation of the CNN-PdM framework can be performed for the usage of data from multiple sources. All the implementation of this work was made using python 3. The following libraries were used: *numpy*, *pandas*, *Keras* + *tensorflow*, *scikit-learn* (*sklearn*), *requests*, *time*, and *json*.

This chapter also provides the evaluation of the proposed framework. A description of the data used in each one of the case studies used in this research and also the evaluation of the CNN-PdM framework. The data used in this research has been collected in the Claudette MacKay-Lassonde Pavilion (CMLP) at Western University, London, Canada. The data were collected by two WebMeters 36F, an accurate equipment able to capture real-time energy data and which configuration is able to accommodate single, dual, and three phase assets. The Webmeters were connected to 8 assets from CMLP and their data have been captured. The installation of these meters was possible because of a partnership between Western University and T-innovation Partners (TiP). TiP provides CircuitMeter devices for measuring various electricity attributes and CircuitMonitor software which collects data from the meters, stores

them in the cloud, and presents the received data. TiP works with clients to lower their electricity cost through the use of innovative products for monitoring and controlling of electrical systems.

The data are derived from two assets: Supply Fan 103B (SF103B) and Return Fan 101 (RF101). These assets were selected because they present very different electrical patterns from each other. SF103B presents an average energy consumption of 8.31 kW/h and average amps of 8.67 A, while RF101 presents 0.44 kW/h and 0.54 A average values for energy consumption and amps, respectively. 1 year of hourly data was collected from February 1, 2018 to January 31, 2019.

4.1 Implementation

This section presents the implementation of the CNN-PdM framework. Details regarding the libraries (libs) used in each one of the stages and steps are provided in this section.

4.1.1 Data Collection

For this stage, libraries (libs) such as *json, request* and *time* were used. The lib *requests* was used to perform multiple API requests. *time* was used to sleep the thread for 1 second before the next request due to an API's constraint. The result of the request is stored in a json file, adding also the date of the requested day.

4.1.2 Data Preprocessing

This stage contains 4 steps: Dataset Construction, Data Imputation, Feature Creation, and Normalization. The libs used in each one of the stages are provided with details of their usage.

Dataset Construction

After all the data are gathered in a single file, the **Dataset Construction** module is responsible for grouping all the features. If external data is used, it must be combined with the asset's data by this module. The implementation was made by using *json* lib to read each line of the file and convert them into json objects. A parser was created to read each row and create entries into a *pandas* dataframe that contained information regarding the measurement name, measurement value, and date and time (timestamp). The following step was to group the feature values and name by timestamp.

Data Imputation

The **Data Imputation** module is responsible for filling all the missing values. In this research, linear interpolation was used in each column to perform this step. Linear interpolation was used because it is a method accurate enough if the values are closely spaced (which is the case in this research). The *pandas* dataframe object has a method *interpolate* that was used to perform the interpolation for each one of the columns using the method 'linear'.

Feature Creation

In this research, features are created from the timestamp to add temporal information, such as information regarding the day (if it is a weekday or a weekend), hour, and month. This step is essential because the temporal features add relevant information, such as seasonality.

To create the information regarding the type of the day (if it is a week-day or weekend day) from the timestamp, it was only needed to access the date part. At *pandas* dataframe, the *datetime* object contains information regarding the number of the *day of week* (0 represents Monday and 6 Sunday). The information regarding the month (month_cos and month_sin) were obtained using the Equations 3.1 and 3.2. The information regarding the hour of the day (hour_cos and hour_sin) follows an analogue pattern, shown in Equations 3.3 and 3.4, presented in Section 3.2.3.

Normalization

After all the data are together, the **Normalization** module used the *MinMax Scaler* to perform normalization. To do so, the *sklearn* method *MinMaxScaler* was used and the normalization interval was (-1, 1).

4.1.3 Data Transformation

For the **Data Transformation** stage, two approaches were tested: Gramian Angular Field (GAF), presented in Section 2.1.3, and Multiplication, introduced in 3.3.

The feature transformation is performed by using either a method to perform the multiplication of an array by each one of the elements, resulting in a matrix instead of the array or a method to transform the data using GAF is used. This process is applied to the entire input dataset by using the method *apply_along_axis* from *numpy* lib.

An example of this transformation through multiplication can be seen in Figure 4.1. It is possible to observe that the subtraction makes the image too dark if one compares the transformation by GAF (Figure 2.2) with the data transformed by multiplication (Figure 4.1). The problem with a very dark image is that it becomes more homogeneous (more black pixels), therefore the CNN model is not able to detect the change of pattern in the homogeneous region. Both images are transformations applied over the same example of the SF103 asset's dataset.

4.1.4 CNN Model Creation

This stage contains 3 steps: CNN Model Parameter Tuning, CNN Model Training and CNN Model Testing.

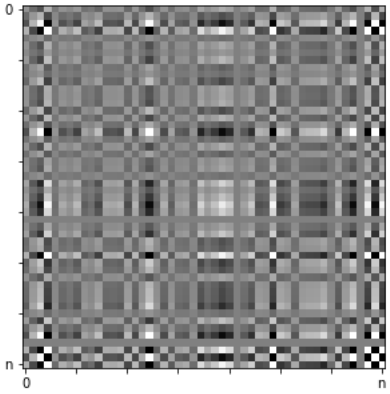

Figure 4.1: An example of sensor data represented as an image (multiplication).

CNN Model Parameter Tuning

The **CNN Model Parameter Tuning** works in the following flow: the dataset is randomly sampled in a 5-fold cross-validation schema, splitting 80% for training and 20% for testing. Then the parameters are randomly combined (otherwise it would be an exhaustive search) and the 5-fold cross-validation sets are used at this step. Accuracy (Eq. 2.1) is the metric used to evaluate how well a parameter combination performs. This step produces an optimized set of parameters for the model.

In order to perform the random sampling, the method *train_test_split* from *sklearn* is used. To perform the randomized search, an object of the *RandomizedSearchCV* class from *sklearn* is created. As the PdM problem, in this case, is considered a classification one and there are features to capture the seasonality, the data shuffle causes no drawback. This randomized sampling approach was also used by Leathy *et al.* [39]. As parameters we used keras lib to create a *Sequential* model using a wrapper from *keras* to *sklearn* lib. This wrapper is responsible for creating the CNN model using as input a set of hyperparameters (number of layers, number of kernels, kernel size, number of epochs and so on). The *RandomizedSearchCV* object is used to fit the training test input to the output with a list of possible hyperparameters. After the fit is finished, it is possible to access the hyperparameters combination that provided the better

results.

CNN Model Training

In this research, 30 simulations considering both **CNN Model Training** and **CNN Model Testing** are made for each experiment, considering 80% of the data as training set and the remaining 20% as testing set, respectively. These sets are randomly sampled from the whole dataset, and this data split process is repeated for each one of the simulations.

The data are split using the *train_test_split* method and using the best parameters from the previous step, the CNN model is created through the wrapper function and the fit method is used to train the data.

CNN Model Testing

The test set created in the previous step is evaluated in the **CNN Model Testing** by calculating four metrics described in Section 3.4.3 and also the confusion matrix. The sklearn lib has a package with numerous test functions. In this work, *accuracy*, *precision*, *recall* and *f1_score* were used. The *confusion_matrix* was also calculated for each experiment.

4.2 Evaluation

This section provides information common to both case studies. The data from both assets contain the same features. The steps used in both case studies are the ones presented in Chapter 3.

4.2.1 Data Collection

The data are acquired using an API (Application Programming Interface) that has no concept of raw data. Therefore, data are considered a combination of **datapoint** and **operator**. The

datapoints collected are described in Table 4.1. With advice from Western's Facilities Management, data from a specific asset (SF103 - Supply Fan 103) have been used in the experiments. Another asset (RF101 - Return Fan 101) was selected because they offer diverse data; therefore, it is important to evaluate if the CNN-PdM framework is able to perform well in both scenarios.

Table 4.1: List of available datapoints.

Feature	**Description**
Current	in A
Energy Consumption	in kWh
Power Factor	how efficiently the device is working
Outside Humidity	%
Outside Temperature	in Celcius
Reactive Power	power absorbed by a circuit
Real Power	power dissipated by a circuit
Apparent Power	sum of reactive and real power
Impedance	in ohms

Over the datapoints presented in Table 4.1, it is necessary to apply one of the operators presented in Table 4.2. These operators are applied according to a given time window. For this work, the time window was 1 hour. From the combination of datapoints and operators, the sum of the energy consumption for a given hour, or the maximum value of the current for the last hour, are examples of possible values that can be requested using the API. For both case studies, 1 year of data were collected from February 1, 2018 to January 31, 2019.

Table 4.2: List of available operators.

Operator	**Description**
avg	Average value
sum	Sumation of the values
max	Maximum value
min	Minimum value
sd	Standard Deviation

4.2.2 Data Preprocessing

This stage contains 5 steps: Dataset Construction, Data Imputation, Feature Creation, Anomaly Creation, and Normalization. The importance of this stage is to improve the quality of the data that will be fed into the CNN models.

Dataset Construction

After all the data are gathered in a single file, the **Dataset Construction** step is responsible for grouping all the features. A parser was created to perform this task. The parser groups all the features from the previous stage by date and time (timestamp). The output of the parser is a large matrix containing the timestamp plus the 45 possible features.

Data Imputation

The **Data Imputation** is performed using linear interpolation to fill the missing values for each one of the available features. Linear interpolation was used because it is a method accurate enough if the values are closely spaced (which is the case in this research).

Feature Creation

As soon as the missing values are filled, this step is responsible for using the date (timestamps) column to create five more contextual features. The timestamp contains two types of information: date and time. From the date, it is possible to determine if the example is a weekday or not. It is also possible to add information about the month of the year (two new features are generated - $sin(\frac{month}{12})$ and $cos(\frac{month}{12})$, which provide knowledge regarding seasonality. From the time, two more features are created: $sin(\frac{hour}{24})$ and $cos(\frac{month}{24})$ [46]. Every feature that uses sine and cosine to perform a smooth transition from one day to another (in the case of features created from the hour) or one year to another (in the case of features created from the date). The feature creation step is important since it adds more temporal information to the data.

Anomaly Creation

To evaluate the framework, anomalous data had to be created. When the building is new, or the sensors have only been recently installed, faulty data are not available. The anomalous data emulates the existence of faulty data, allowing the usage of supervised learning techniques. It also helps in scenarios when the dataset is **unbalanced**. A dataset is known as unbalanced if the interest class represents a small portion of the data. This may occur in cases which it is rare to observe assets failing. This lack of faulty data may cause traditional ML techniques to perform poorly [47]. The creation of anomalous data can help with problems with small and unbalanced datasets.

One assumption made in this research is that the device does not present any faulty data. As the WebMeters 36Fs were recently installed and no information regarding faulty data was provided, abnormal data must be created. This step generates anomalies by combining available features, in this case, electrical features, with different contextual features shifted by 9 hours. Another assumption made was that the University is more crowded from 9 am to 6 pm (9 hours interval), so it seemed reasonable to have a 9 hours shift to combine electrical and contextual features. Doing this makes it possible to reproduce a higher energy consumption with a crowded building to a time in which a reduced number of people should be there. A pattern that is expected to happen on a Monday at 1 pm, for instance, with plenty of people in the building, would be simulated to occur on Monday at 10 pm by mixing the electrical features from 1 pm with temporal features from 10 pm. The faulty data are added to the previous dataset. The normal data were labeled with 0, while the anomalous data were labeled with 1.

Normalization

The **Normalization** step is performed to normalize the data in the $[-1, 1]$ interval, accordingly to the Equation 3.5. This step is important to bring all the data to the same range, which can make the model less sensitive to bias.

4.2.3 Data Transformation

Two data transformation methods were tested: multiplication and GAF. The GAF was used before to perform financial time-series analysis by Chen *et al.* [27]. This approach was presenting dark and homogeneous images, which could be a problem for CNNs (Figure 2.2), since CNNs rely on learning different patterns from shapes, and a homogeneous image would present a smaller number of forms to be learned. The multiplication method, proposed in this research, was able to represent the data in a more heterogeneous way (Figure 4.1). An example of the difference between multiplication and GAF is here presented: given two features $energy\_consumption - avg = 0.1$ and $amps - sd = 0.2$, for instance, transformed using GAF outputs -0.9548, while transformed using the multiplication method outputs 0.02. Therefore, the GAF output is much darker than the multiplication output.

4.2.4 CNN Model Creation

The **CNN Model Creation** stage contains 3 steps. The CNN Parameter Tuning occurs by selecting the most suitable set of hyperparameters. With possession of the most suitable hyperparameters, 30 simulations are performed. For each one of the simulations the data is randomly split in two sets: 80% is used in the training step (CNN Model Training) and 20% used in the testing step (CNN Model Testing).

4.3 Case Study 1: Supply Fan 103B (SF103B)

The SF103B asset was selected after meetings with the Facilities Management department from Western University. It was described as the most important asset to monitor. SF103B presents an average energy consumption of 8.31 kW/h and average amps of 8.67 A. Figure 4.2 presents how these two features behave from February 1, 2018 to January 31, 2019. This dataset was used for all experiments performed in this Case Study. Some features presented missing data. The number of missing values for each one of these features are presented in Table 4.3.

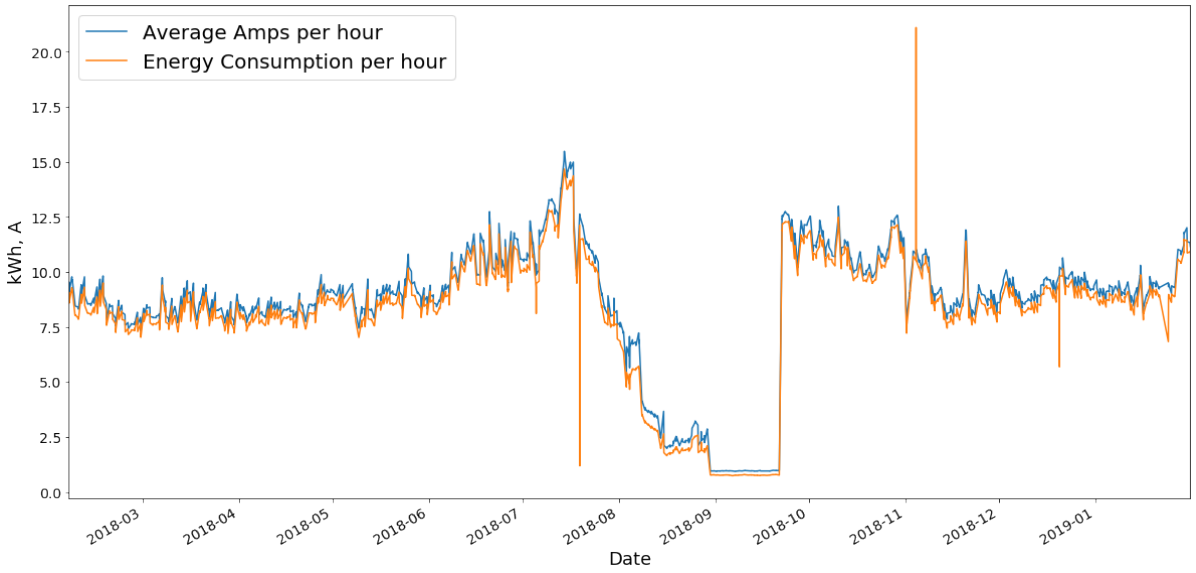

Figure 4.2: Amps and energy consumption - SF103B.

Some features presented missing data. The number of missing values for each one of these features are presented in Table 4.3.

Regarding the data presented in Figure 4.2, it is possible to observe that from June, 2018 until mid July, 2018, it starts a trend of increase, then it drastically decreases and reaches its lower values by September, 2018. Although this is an unusual pattern, it is not possible to affirm that it is a failure state mainly because the WebMeter 36F never presented a 0A or 0kWh measurement. There are 3 other supply fans in the building, therefore, this change in the behavior may be just a load distribution among the assets.

4.3.1 Evaluation

This section presents the evaluation of the CNN-PdM framework. Two experiments were performed using different data transformation methods: Multiplication and GAF.

Convolutional Neural Network

Multiplication and GAF feature transformation methods were tested. After the feature transformation, one must generate an optimized architecture of the CNN. Each convolutional (conv)

Table 4.3: Number of missing values for each one of the features - SF103B.

Number of Missing Values	**Features**
3	Outside Temperature - min Outside Temperature - sd
10	Amps - min Amps - sd Apparent Power - min Apparent Power - sd Energy Consumption - min EnergyConsumption - sd PowerFactor - min PowerFactor - sd Impedance - max Impedance - min Impedance - sd Impedance - avg Impedance - sum

layer is followed by a dropout layer. After a sequence of convs and dropouts, one flattened layer is used to gather the information of all the filters from the last conv layer, and a final neuron which would represent the example's class. Six parameters are tuned for this approach:

- number of conv layers (n_hidden_layers): from 1 to 5
- number of filters (n_filters): 15, 20, 25, 30, 35, 40, or 45
- kernel size (kernel_size): from 3 to 8
- number of epochs (n_epochs): 100, 150, 200, 250, 300, 350
- learning rate (lr): from 10^{-4} to $5 \cdot 10^{-3}$
- optimizer: sgd, adam, or adadelta

The number of conv layers was selected based on the number of examples and number of filters. Higher number of conv layers would represent more neurons (and also higher number of filters and kernel sizes). Based on the number of examples available (only 1 year of data), it

does not seem feasible to have CNNs with more than 5 conv layers. In fact, Table 4.4 shows that 3 conv layers presented the best results. The batch size parameter is fixed to 100, the dropout rate is fixed to 20% of the neurons, and the loss function used is binary cross entropy. Cheng *et al.* [48] performed experiments aiming at the impact of batch size and number of threads for the serving latency of recommender systems. They showed that there is almost no difference from 100 to 50 examples in the batch size. The dropout rate was selected after literature review [49]. The number of conv layers could be increased if more data was available. The learning rate range, which is the most important parameter, was picked taking in consideration courses made by Jeremy Howard, from fast.ai [50]. In the second lesson of his course, he explained that the default learning rate used in fast.ai library is $3 \cdot 10^{-3}$ because it works fine for most of the cases; therefore, a range around this value was selected. Regarding the optimizers, Adam is an optimization of SGD (Stochastic Gradient Descent) that considers momentum, i.e. a way to take into consideration previous examples of the batch. The most suitable parameters found by random search for both feature transformation methods are presented in Table 4.4. After model optimization, 30 simulations are performed as described in Section 4.1.4.

Table 4.4: Parameters for CNN_MULT and CNN_GAF experiments - SF103B.

Parameter	**Value**
n_hidden_layers	3
n_filters	45
kernel_size	8
n_epochs	350
lr	$3 \cdot 10^{-3}$
optmizer	sgd

The *CNN_GAF* experiment presented an overall accuracy slightly over 0.933, while the *CNN_MULT* approach presented the best results for each one of the metrics evaluated, achieving slightly over 0.953 accuracy, presented in Table 4.11. This difference represents an improvement of 2% (2.08%) over the *CNN_GAF* experiment. The lower sd (presented in Table 4.12) implies that the solution is reliable and that there is little difference between the 30 simulations of this experiment.

Comparison

In this research, six more experiments have been performed for each asset to compare with the ones that used CNN. Three different ML techniques were considered: SVM, RF, and MLP; these experiments were performed with and without considering feature selection as an extra step (6 experiments).

I **Random Forest:**

Two different experiments were performed using random forest: one with and another one without considering a feature selection step. Since Random Forest computes the features' importance, it is possible to select which features are more relevant to the problem. The features could be selected based on their importance for the RF model; the features that presented importance greater than $0.8 \cdot$ *average* feature importance were selected. The selected features were: hour (cos), hour (sin), weekend, outside humidity (max), outside humidity (sum), outside temperature (max), outside temperature (sum), power factor (min), power factor (sum), reactive power (max), reactive power (min), reactive power (sd), and reactive power (sum). The selected features present a combination of contextual features describing the environment, power factor, which represents how efficient the equipment is working, and reactive power, which represents the voltage drop caused by inductors and capacitors.

After performing the feature selection process, the stage of RF model tuning begins. The random search algorithm considers statistical distribution over the possible parameters and the algorithm uses accuracy as a score function. Again, the data is randomly split, and the maximum depth (max_depth) of the trees and the number of estimators (n_trees) are selected:

- max_depth: from 5 to 15
- n_trees: 6, 7, 8, 9, 10, 15, 20, or 25

Table 4.5 presents the best parameters for the RF experiment without feature selection (RF_NFS). The experiment that considers feature selection (RF_FS) presented the parameters shown in Table 4.6 as the best ones. After the most suitable parameters were found, 30 simulations are performed and evaluated as described in Section 4.1.4.

Table 4.5: Parameters for RF_NFS experiment - SF103B.

Parameter	**Value**
max_depth	8
n_trees	13

Table 4.6: Parameters for RF_FS experiment - SF103B.

Parameter	**Value**
maximum depth	25
number of trees	12

From Table 4.11 it is possible to observe that the *RF_NFS* experiment presented an overall accuracy above 0.835 for the test examples. Since recall (Equation 2.3) presents a higher value, it is possible to infer that the normal data are correctly classified for most of the cases. It is also possible to observe that the *RF_FS* presents an improvement if compared with the *RF_NFS* experiment. It's accuracy is above 0.873 for the test examples. The accuracy was increased by over 4% when compared to the previous experiment. Since the precision presents a higher value, one can infer that only 10% of the examples classified as normal were false positives.

II **Support Vector Machine:**

Similarly to the Random Forest experiments, two experiments were conducted, with and without considering feature selection as an extra step. When considering feature selection, the same features selected by the Random Forest experiment were used, as the SVM algorithms do not provide information about feature importance. Two parameters were selected to tune:

- kernel: *rbf*, *polynomial*, or *sigmoid*
- coef0: from 0.05 to 1.00 with increments of 0.5

The most suitable parameters found for the SVM without considering feature selection (*SVM_NFS*) are presented in Table 4.7, while Table 4.8 shows the best parameters for the SVM considering feature selection (*SVM_FS*). After the ideal parameters were found, the 30 simulations are performed and evaluated as described in Section 4.1.4.

Table 4.7: Parameters for SVM_NFS experiment - SF103B.

Parameter	**Value**
kernel	rbf
coef0	0.85

Table 4.8: Parameters for SVM_FS experiment - SF103B.

Parameter	**Value**
kernel	rbf
coef0	0.50

The *SVM_NFS* achieved the best result so far, besides the CNN ones, for all the evaluated metrics: around 5% improvement on overall accuracy (0.921) when compared with *RF_FS* experiment. The presented results were slightly worst (0.869) when comparing to the *RF_FS* experiment. These results are presented in Table 4.11.

III **Multi-Layer Perceptron:**

Similarly to SVM and RF, two experiments were conducted, considering feature selection or not as another stage of the process. Five parameters were selected to be tuned:

- n_layers: from 1 to 7
- n_neurons: 15, 20, 25, 30, 35, 40
- n_epochs: 100, 200 or 300
- lr: from 10^{-4} to $5 \cdot 10^{-3}$

- optimizer: sgd, adam, adadelta

The most suitable parameters found for this problem without considering feature selection (*MLP_NFS*) are presented in Table 4.9, while the most suitable parameters when using feature selection (*MLP_FS*) are shown in Table 4.10. After the model tuning for this experiment, the 30 simulations are performed as described in Section 4.1.4.

Table 4.9: Parameters for MLP_NFS experiment - SF103B.

Parameter	**Value**
n_layers	5
n_neurons	30
epochs	300
lr	10^{-3}
optmizer	adam

Table 4.10: Parameters for MLP_FS experiment - SF103B.

Parameter	**Value**
n_layers	2
n_neurons	35
epochs	300
lr	$2 \cdot 10^{-3}$
optmizer	adam

The *MLP_NFS* experiment presented a 1.5% increase on the accuracy when compared to the *SVM_FS*, as can be observed in Table 4.11. This experiment achieved the second best result (0.935). On the other hand, the *MLP_FS* presented almost 0.869 accuracy. The effect of the RF feature selection for the MLP was disastrous, even worse than for the SVM experiment. The accuracy decreased more than 9%, as can be observed in Table 4.11.

4.3.2 Discussion

This section presents the results from eight different experiments. Tables 4.11 and 4.12 show the average and standard deviation for each experiment. Figure 4.3 presents the same results

that are presented in Table 4.11. However, both are presented for a matter of better visualization that the graph offers. Table 4.13 presents the confusion matrix for each one of the experiments.

As can be seen in all three tables, *CNN_MULT* presents better results compared to the other approaches. The *CNN_MULT* approach outperforms the conventional ML techniques because of the conv layers' capability of extracting information automatically. The *CNN_MULT* outperform the *CNN_GAF* approach probably because of the absence of the subtraction term from GAF. The transformation by multiplication does not generate a very dark image, and it helps the CNNs capability to extract data representation.

Among the Random Forest experiments, the *RF_FS* presents better results (a 4% improvement if compared with the *RF_NFS*), which shows that the feature selection method works fine with this technique. With fewer features, the model could learn with a lower level of confusion, allowing the model to perform a lower number of errors.

For the SVM experiments, it is possible to see that the features selected using Random Forest features' importance did not work well, as the *SVM_NFS* presented better results than the *SVM_FS* experiment (0.921 and 0.869 accuracy, respectively). This probably happens because SVM and RF handle data differently from each other.

It is possible to observe that the same effect happens in the MLP experiments. The *MLP_NFS* experiment was able to extract the relationship among the features better than the RF and SVM approaches. The MLP_NFS results (0.935) are comparable to the *CNN_GAF* (0.933), as their accuracy difference is lower than 0.16%. If one must choose between them, the MLP approach is preferable because it is a simpler and faster model to train and run. As the *CNN_MULT* outperformed the other models, it should be used to generate warnings advising that an asset needs maintenance.

In order to evaluate if the results for each one of the experiments are statistically different, two tests were performed: Kruskal-Wallis H-test for independent samples and Wilcoxon signed-rank test [51]. The Kruskal-Wallis test is applied considering all the 8 experiments at once. If they present a p-value lower than 0.05, the Wilcoxon test should be performed consid-

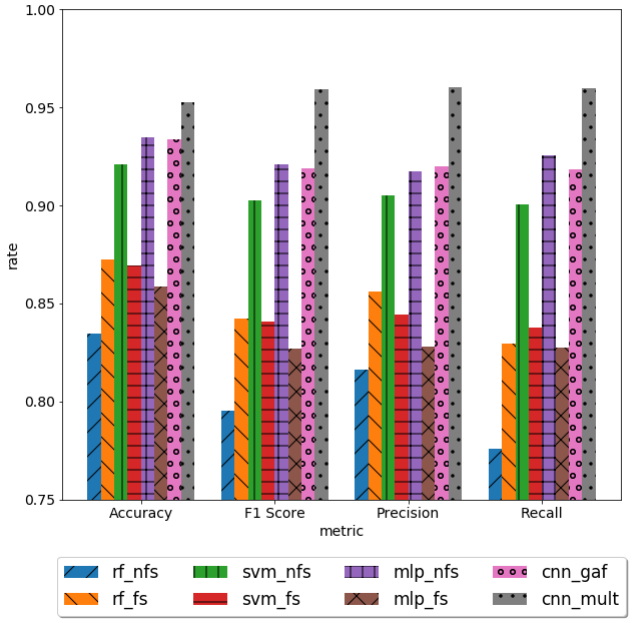

Figure 4.3: Comparison among experiments - SF103B (average values).

ering each pair of experiments. The Kruskal-Wallis test presented a p-value of $9.2576 \cdot 10^{-44}$, therefore the Wilcoxon test was needed. Its results are shown in Table 4.14. Since there are 28 possible combinations of experiments, the Bonferroni correction [52] must be used. In this scenario, the p-value to represent statistically different results should be $\frac{0.05}{28} = 0.0018 = 1.8 \cdot 10^{-3}$. The only experiments that presented non-statistically difference were (SVM_NFS, MLP_NFS), with a p-value equals to 0.3085. Therefore, the results obtained from the CNN_MULT experiment are statistically different from the others. This leads to the conclusion that they represent an improvement.

Table 4.11: Metrics comparison among experiments - SF103B (average % values).

	Accuracy	**F1 Score**	**Precision**	**Recall**
RF_NFS	83.4626	79.5351	81.6448	77.5802
RF_FS	87.2758	84.2374	85.6175	82.9389
SVM_NFS	92.0833	90.2731	90.5198	90.0451
SVM_FS	86.9225	84.1037	84.4617	83.7828
MLP_NFS	93.4705	92.1019	91.7214	92.5619
MLP_FS	85.8492	82.7033	82.7834	82.7454
CNN_GAF	93.3228	91.9225	92.0055	91.8577
CNN_MULT	**95.2611**	**95.9129**	**96.0051**	**95.9558**

Table 4.12: Metrics comparison among experiments - SF103B (sd % values)

	Accuracy	**F1 Score**	**Precision**	**Recall**
RF_NFS	1.9718	2.4035	2.6267	2.9241
RF_FS	0.8050	1.1554	1.4916	1.9503
SVM_NFS	1.0131	1.2314	1.4912	1.5589
SVM_FS	1.5464	1.7964	2.1787	2.1514
MLP_NFS	1.0369	1.2620	2.4000	2.0861
MLP_FS	1.1297	1.5687	2.9023	2.6512
CNN_GAF	0.9188	1.1523	1.5812	1.3691
CNN_MULT	**0.6155**	**0.7046**	**1.2487**	**1.2196**

Table 4.13: Confusion Matrix for all experiments - SF103B.

	Predicted: 0	Predicted: 1	
Actual: 0	RF_NFS: 648 RF_FS: 672 SVM_NFS: 698 SVM_FS: 661 MLP_NFS: 698 MLP_FS: 656 CNN_GAF: 697 CNN_MULT: 712	RF_NFS: 91 RF_FS: 72 SVM_NFS: 49 SVM_FS: 79 MLP_NFS: 44 MLP_FS: 89 CNN_GAF: 41 **CNN_MULT: 30**	RF_NFS: 739 RF_FS: 744 SVM_NFS: 747 SVM_FS: 740 MLP_NFS: 742 MLP_FS: 745 CNN_GAF: 738 CNN_MULT:742
Actual: 1	RF_NFS: 117 RF_FS: 88 SVM_NFS: 51 SVM_FS: 85 MLP_NFS: 39 MLP_FS: 89 CNN_GAF: 43 **CNN_MULT: 30**	RF_NFS: 406 RF_FS: 430 SVM_NFS: 464 SVM_FS: 437 MLP_NFS: 481 MLP_FS: 428 CNN_GAF: 481 CNN_MULT: 490	RF_NFS: 523 RF_FS: 518 SVM_NFS: 515 SVM_FS: 522 MLP_NFS: 520 MLP_FS: 517 CNN_GAF: 524 CNN_MULT: 520
	RF_NFS: 765 RF_FS: 760 SVM_NFS: 749 SVM_FS: 746 MLP_NFS: 737 MLP_FS: 745 CNN_GAF: 740 CNN_MULT: 742	RF_NFS: 497 RF_FS: 502 SVM_NFS: 513 SVM_FS: 516 MLP_NFS: 525 MLP_FS: 517 CNN_GAF: 522 CNN_MULT: 520	

Table 4.14: p-values among experiments - SF103B.

p-value	pair
$1.7311 \cdot 10^{-6}$	(MLP_FS, CNN_GAF) (SVM_NFS, CNN_MULT) (RF_NFS, CNN_MULT) (RF_NFS, MLP_NFS) (RF_NFS, SVM_NFS) (SVM_FS, CNN_MULT) (SVM_FS, CNN_GAF) (SVM_FS, MLP_NFS) (SVM_NFS, SVM_FS) (RF_FS, CNN_MULT) (RF_FS, SVM_NFS) (RF_NFS, CNN_GAF) (RF_NFS, SVM_FS) (MLP_NFS, CNN_MULT) (MLP_NFS, MLP_FS) (SVM_NFS, MLP_FS) (RF_FS, MLP_NFS) (MLP_FS, CNN_MULT) (RF_FS, CNN_GAF)
$2.3505 \cdot 10^{-6}$	(RF_NFS, RF_FS)
$3.8991 \cdot 10^{-6}$	(SVM_FS, MLP_FS)
$4.5502 \cdot 10^{-6}$	(SVM_NFS, CNN_GAF)
$4.7886 \cdot 10^{-6}$	(CNN_GAF, CNN_MULT)
$5.7697 \cdot 10^{-5}$	(RF_NFS, MLP_FS)
$1.4767 \cdot 10^{-4}$	(MLP_NFS, CNN_GAF)
$3.8783 \cdot 10^{-4}$	(RF_FS, MLP_FS)
$5.7007 \cdot 10^{-4}$	(RF_FS, SVM_FS)
0.3085	(SVM_NFS, MLP_NFS)

4.4 Case Study 2: Return Fan 101 (RF101)

The RF101 asset was selected because it was the asset which electric profile varies the most if compared with the SF103B. The goal of selecting this asset's data was to show that the CNN-PdM framework is capable of handling the predictive maintenance problems with different assets. RF101 presents 0.44 kW/h and 0.54 A average values of energy consumption and amps, respectively. Figure 4.4 presents how these two features behaved from February 1, 2018

Table 4.15: Number of missing values for each one of the features - RF101.

Number of Missing Values	**Features**
1	Amps (avg, max, sum) Apparent Power (avg, max, sum) Energy Consumption (avg, max, sum) Outside Humidity (avg, max, sum, sd, min) Outside Temperature (avg, max, sum) Power Factor (avg, max, sum) Reactive Power (avg, max, sum) Real Power (avg, max, sum)
4	Outside Temperature (min, sd)
29	Reactive Power (min, sd)
32	Real Power (min, sd)
45	Amps (min, sd) Apparent Power (min, sd) Impedance (avg, max, min, sd, sum)
48	Power Factor (min, sd) Energy Consumption (min, sd)

to January 31, 2019. Some features presented missing data. The number of missing values for each one of these features are presented in Table 4.15.

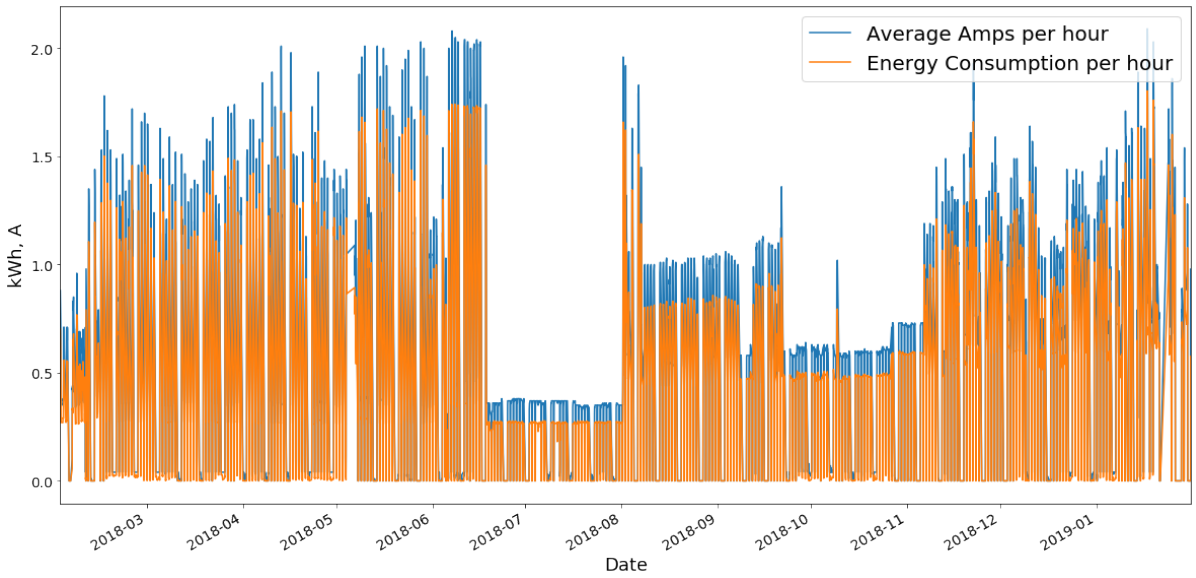

Figure 4.4: Amps and energy consumption - RF101.

4.4.1 Evaluation

This section presents the evaluation of the CNN-PdM framework. The same two methods to perform data transformation are tested: Multiplication and GAF.

Convolutional Neural Network

The model tuning performed has the same parameters as the ones tested for the SF103B asset:

- number of conv layers (n_hidden_layers): from 1 to 5
- number of filters (n_filters): 15, 20, 25, 30, 35, 40, or 45
- kernel size (kernel_size): from 3 to 8
- number of epochs (n_epochs): 100, 150, 200, 250, 300, 350
- learning rate (lr): from 10^{-4} to $5 \cdot 10^{-3}$
- optimizer: sgd, adam, or adadelta

The combination that presented the better results are shown in Table 4.16.

Table 4.16: Parameters for CNN_MULT and CNN_GAF experiments - RF101.

Parameter	**Value**
n_hidden_layers	3
n_filters	40
kernel_size	8
n_epochs	350
lr	$2 \cdot 10^{-3}$
optmizer	adam

The CNN using GAF (*CNN_GAF*) experiment presented an overall accuracy slightly over 0.962. The *CNN_MULT* approach presents the best results for each one of the metrics evaluated and achieved 0.981 accuracy, presented in Table 4.23. This difference represents an improvement of 2.00% over the *CNN_GAF* experiment. The lower sd (presented in Table 4.24) implies

that the solution is reliable and that there is little difference between the 30 simulations of this experiment.

Comparison

In this research, six more experiments have been performed for each asset to compare with the ones performed using CNN. Three different ML techniques were considered: SVM, RF, and MLP. These experiments were performed with and without considering feature selection as an extra step (6 experiments).

I **Random Forest:**

As was performed for the SF103BT, two different experiments were performed: one with and another one without considering a feature selection step. The selected features are: amps (avgall), amps (sum), apparent power (min), energy consumption (max), outside temperature (sum), power factor (max), reactive power (max), real power (avgall), real power (sum), weekend, hour (sin), and hour (cos).

The selected features present a combination of contextual, temporal and electrical features. This selection corroborates with the thought that time related features and the environment conditions are important to the PdM problem.

For the model tuning step, the different values of maximum depth and number of trees were tested:

- max_depth: from 5 to 15
- n_trees: 6, 7, 8, 9, 10, 15, 20, or 25

Table 4.17 presents the most suitable parameters for the RF experiment without feature selection (*RF_NFS*). The experiment that considers feature selection (*RF_FS*) presented the parameters shown in Table 4.18 as the most suitable ones. After the ideal parameters were found, the 30 simulations are performed and evaluated as described in Section 4.1.4.

Table 4.17: Parameters for RF_NFS experiment - RF101.

Parameter	Value
max_depth	7
n_trees	14

Table 4.18: Parameters for RF_FS experiment - RF101.

Parameter	Value
max_depth	20
n_trees	6

From Table 4.23 it is possible to observe that the *RF_NFS* experiment presented an overall accuracy above 0.936 for the test examples. The *RF_FS* presents an improvement when compared with the *RF_NFS* experiment. It's accuracy is above 0.954 for the test examples. The accuracy was increased by 1.9% when compared to the previous experiment.

II **Support Vector Machine:**

Similarly to the Random Forest experiments, two experiments were conducted, with and without considering feature selection as another stage of the process. When considering feature selection, the same features selected by the Random Forest experiment were used, as the SVM algorithms do not provide information about feature importance. Two parameters were selected to tune: kernel and coef0 (independent term in kernel function that is used if the kernel is polynomial or sigmoid).

- kernel: *rbf*, *polynomial*, or *sigmoid*
- coef0: from 0.05 to 1.00 with increments of 0.5

The most suitable parameters found for the SVM without considering feature selection (*SVM_NFS*) are presented in Table 4.19, while Table 4.20 shows the parameters for the SVM considering feature selection (*SVM_FS*). After the ideal parameters were found, 30 simulations are performed and evaluated as described in Section 4.1.4.

Table 4.19: Parameters for SVM_NFS experiment - RF101.

Parameter	Value
kernel	rbf
coef0	0.75

Table 4.20: Best parameters for SVM_FS experiment - RF101.

Parameter	Value
kernel	rbf
coef0	0.55

The *SVM_NFS* approach achieved the best result so far, besides the CNN ones, achieving a very similar result to the *RF_FS* experiment: 0.955 accuracy rate. The *SVM_FS* presented results slightly worst than the *SVM_NFS* experiment (0.952). These results are presented in Table 4.23.

III **Multi-Layer Perceptron:**

Similarly to SVM and RF, two experiments were conducted, considering feature selection or not as another stage of the process. Four parameters were selected to be tuned: the number of hidden layers, number of epochs, the number of neurons in each layer and the optimizer. The values for each parameters can be seen below:

- n_layers: from 1 to 7
- n_neurons: 15, 20, 25, 30, 35, 40
- n_epochs: 100, 200 or 300
- lr: from 10^{-4} to $5 \cdot 10^{-3}$
- optimizer: sgd, adam, adadelta

The most suitables parameters found for this problem without considering feature selection (*MLP_NFS*) are presented in Table 4.21, while the parameters selected if using

feature selection (*MLP_FS*) are shown in Table 4.22. After the definition of the best parameters for this experiment, the 30 simulations are performed as described in Section 4.1.4.

Table 4.21: Parameters for MLP_NFS experiment - RF101.

Parameter	**Value**
n_layers	3
n_neurons	35
epochs	300
lr	$1.1 \cdot 10^{-3}$
optmizer	adam

Table 4.22: Parameters for MLP_NFS experiment - RF101.

Parameter	**Value**
n_layers	3
n_neurons	30
epochs	300
lr	$1.4 \cdot 10^{-3}$
optmizer	adam

The *MLP_NFS* experiment presented almost 0.966 accuracy rate, as can be observed in Table 4.23. This experiment achieved the second best result. The *MLP_FS* experiment presented almost 0.947 accuracy. The effect of the RF feature selection for the MLP was not as bad as it was in the SF103B experiment, but it was also worse to have feature selection than not have it in this case.

4.4.2 Discussion

This section presents the results from eight different experiments. Tables 4.23 and 4.24 show the average and standard deviation for each experiment. Figure 4.5 presents the same results that are presented in Table 4.23. However, both are presented for a matter of better visualization that the graph offers. Table 4.25 presents the confusion matrix for each one of the experiments.

The RF101 dataset was easier for every single ML technique when compared to the SF103B dataset. Every ML technique was able to achieve better results probably because the asset is always being turned off, as is possible to observe in Figure 4.4. If one compares Figure 4.4 with Figure 4.2, the one related to the RF101 seems to be more filled. It represents that it is being turned on and off repeatedly. Nevertheless, RF101 behavior seems to be more homogeneous than the SF103B, which facilitates the task of using machine learning on its data.

The *CNN_MULT* experiment presents a better result for accuracy, f1 score, and recall, while *MLP_NFS* presents better precision. Precision (Equation 2.2) calculates the rate of true positives among every example classified as positive. Higher precision means that the smaller number of false alarms were created by *MLP_NFS* experiment. In the predictive maintenance scenario, a lower amount of false negatives should be considered better, therefore, recall (Equation 2.3) is a more important metric. A lower false negative means that fewer anomalies (assets' faults) were not confused as normal behavior. It is possible to observe in Table 4.25 that *CNN_MULT* presented the smaller number of false negatives. The two experiments with lower false negative values are the ones that use CNN.

The *SVM_NFS* experiment outperformed both *RF_FS* and the *MLP_NFS* experiments, while *CNN_GAF* presented similar results to *SVM_NFS* experiment. MLPs can capture the relationship of features using the weights from the neural networks' layers. GAF transformation, in this case, has an issue with the creation of a dark image. Again, the capability of creating a feature representation of CNN combined with the feature transformation by multiplication allows its *CNN_MULT* to outperform other techniques.

In order to evaluate if the results for each one of the experiments are statistically different, two tests were performed: Kruskal-Wallis H-test for independent samples and Wilcoxon signed-rank test [51]. The Kruskal-Wallis test is applied considering all the 8 experiments at once. If they present a p-value lower than 0.05, the Wilcoxon test should be performed considering each pair of experiments. The Kruskal-Wallis test presented a p-value of $2.7760 \cdot 10^{-42}$, therefore the Wilcoxon test was needed. Its results are shown in Table 4.26. Since there are 28

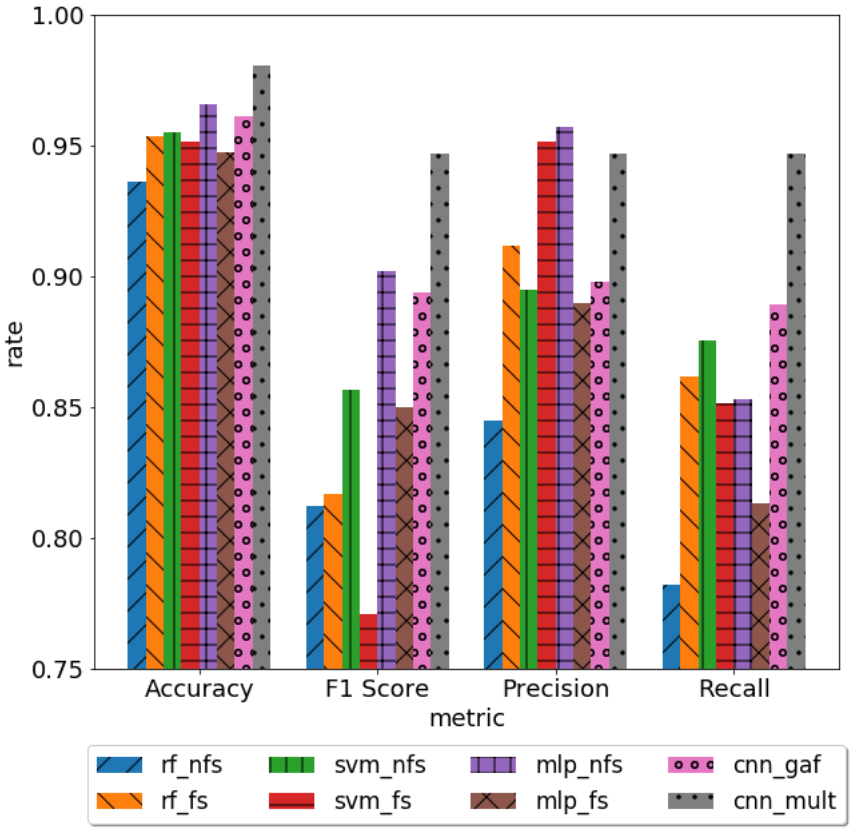

Figure 4.5: Comparison among experiments - RF101 (average values).

possible combinations of experiments, the Bonferroni correction [52] must be used. In this scenario, the p-value to represent statistically different results should be $\frac{0.05}{28} = 0.0018 = 1.8 \cdot 10^{-3}$. The only experiments that presented non-statistically difference were (RF_FS, SVM_FS) and (RF_FS, SVM_NFS), with p-values of $1.3974 \cdot 10^{-2}$ and $2.8485 \cdot 10^{-2}$, respectively. Therefore, the results obtained from the CNN_MULT experiment are statistically different from the others. Consequently, it is possible to conclude that the CNN_MULT outperformed the other approaches, since it presented higher accuracy and recall, and the experiments results are statistically different from each other.

Table 4.23: Metrics comparison among experiments - RF101 (average % values).

	Accuracy	**F1 Score**	**Precision**	**Recall**
RF_NFS	93.6133	81.2339	84.4920	78.2178
RF_FS	95.3631	81.6831	91.1602	86.1619
SVM_NFS	95.5381	85.6459	89.5000	87.5306
SVM_FS	95.1881	77.0731	95.1807	85.1752
MLP_NFS	96.5879	90.2255	**95.7447**	85.3080
MLP_FS	94.7506	85.0000	89.0052	81.3397
CNN_GAF	96.1504	89.3720	89.8058	88.9423
CNN_MULT	**98.0752**	**94.7115**	94.7115	**94.7115**

Table 4.24: Metrics comparison among experiments - RF101 (sd % values).

	Accuracy	**F1 Score**	**Precision**	**Recall**
RF_NFS	1.4137	1.6826	17.5461	17.8435
RF_FS	1.1764	1.3545	1.6743	16.5196
SVM_NFS	1.2743	1.4531	1.3576	1.38843
SVM_FS	1.3419	1.5348	1.8261	1.6716
MLP_NFS	0.8764	1.5483	1.3592	1.4152
MLP_FS	0.9746	1.2687	1.3610	1.2364
CNN_GAF	0.7176	0.7514	0.8145	0.7651
CNN_MULT	**0.5735**	**0.6231**	**0.6231**	**0.6231**

4.5 Summary

This chapter presented the experiments used to validate the CNN-PdM framework. The results show that the CNN-PdM framework with the multiplication data transformation method outperformed all other approaches, achieving almost 0.953 accuracy rate for the SF103B's dataset and over 0.980 accuracy rate when using the RF101's dataset. This proves that the CNN-PdM framework is a reliable method to deal with predictive maintenance.

Table 4.25: Confusion Matrix for all experiments - RF101.

	Predicted : 0	Predicted: 1	
Actual: 0	RF_NFS: 929 RF_FS: 925 SVM_NFS: 913 SVM_FS: 930 MLP_NFS: 924 MLP_FS: 913 CNN_GAF: 914 CNN_MULT: 924	RF_NFS: 29 RF_FS: 16 SVM_NFS: 21 **SVM_FS: 8** **MLP_NFS: 8** MLP_FS: 21 CNN_GAF: 21 CNN_MULT: 11	RF_NFS: 958 RF_FS: 941 SVM_NFS: 934 SVM_FS: 938 MLP_NFS: 932 MLP_FS: 934 CNN_GAF: 935 CNN_MULT: 935
Actual: 1	RF_NFS: 44 RF_FS: 37 SVM_NFS: 30 SVM_FS: 47 MLP_NFS: 31 MLP_FS: 39 CNN_GAF: 23 **CNN_MULT: 11**	RF_NFS: 158 RF_FS: 165 SVM_NFS: 179 SVM_FS: 158 MLP_NFS: 180 MLP_FS: 170 CNN_GAF: 185 CNN_MULT: 197	RF_NFS: 202 RF_FS: 202 SVM_NFS: 209 SVM_FS: 205 MLP_NFS: 211 MLP_FS: 209 CNN_GAF: 208 CNN_MULT: 208
	RF_NFS: 973 RF_FS: 962 SVM_NFS: 943 SVM_FS: 960 MLP_NFS: 955 MLP_FS: 942 CNN_GAF: 937 CNN_MULT: 935	RF_NFS: 187 RF_FS: 181 SVM_NFS: 200 SVM_FS: 166 MLP_NFS: 188 MLP_FS: 191 CNN_GAF: 206 CNN_MULT: 208	

Table 4.26: p-values among experiments - RF101.

p-value	pair
$1.7344 \cdot 10^{-6}$	(RF_NFS, RF_FS) (CNN_GAF, CNN_MULT) (MLP_FS, CNN_MULT) (MLP_NFS, CNN_MULT) (MLP_NFS, MLP_FS) (SVM_FS, CNN_MULT) (SVM_FS, MLP_NFS) (SVM_NFS, CNN_MULT) (SVM_NFS, MLP_NFS) (RF_FS, CNN_MULT) (RF_FS, MLP_NFS) (RF_NFS, CNN_MULT) (RF_NFS, CNN_GAF) (RF_NFS, MLP_FS) (RF_NFS, MLP_NFS) (RF_NFS, SVM_FS) (RF_NFS, SVM_NFS)
$3.5152 \cdot 10^{-6}$	(SVM_FS, CNN_GAF)
$3.8821 \cdot 10^{-6}$	(SVM_NFS, MLP_FS)
$4.2856 \cdot 10^{-6}$	(RF_FS, CNN_GAF)
$8.4660 \cdot 10^{-6}$	(SVM_NFS, CNN_GAF)
$4.4493 \cdot 10^{-5}$	(MLP_NFS, CNN_GAF)
$5.3069 \cdot 10^{-5}$	(RF_FS, MLP_FS)
$1.1494 \cdot 10^{-4}$	(SVM_NFS, SVM_FS)
$1.8909 \cdot 10^{-4}$	(SVM_FS, MLP_FS)
$1.3974 \cdot 10^{-2}$	(RF_FS, SVM_FS)
$2.8485 \cdot 10^{-2}$	(RF_FS, SVM_NFS)

Chapter 5

Conclusions and Future Work

This chapter presents a summary of the conclusions achieved trough the implementation and evaluation of the Convolutional Neural Network Predictive Maintenance framework (CNN-PdM) presented in this research. Possible future works for the CNN-PdM are also provided.

5.1 Conclusions

This work presents the CNN-PdM framework and a novel method to perform data transformation. Since finding the ideal subset of features is a challenge in ML and the number of available features is increasing over time with the advance of IoT, CNNs can be used to tackle these issues; however, to exploit the CNN potential, a data transformation method is also proposed in this work. The multiplication method converts data from 1-D to 2-D, allowing the user to benefit from the CNN ability to perform feature extraction and decrease the amount of previous feature engineering needed.

Accordingly to Lin and Tsen [10], almost 99% of the equipment failures present some sort of indicators that can be tracked to determine if the asset is not working properly. Therefore, the CNN-PdM framework proposed in this research is able to detect indicators that precede failures and advise for maintenance. In the cases of study presented in Section 4.3 for two exhaust fans from CMLP building at Western University, the proposed framework combination

of CNN with multiplication as data transformation method outperformed the others approaches and presented results at least 1.5% better than the second best approach. The contributions of this work are listed below:

- PdM is a maintenance approach in which the asset's health is continuously monitored. This research proposes the usage of CNN to monitor the asset's health status; based on it, a decision can be made regarding if an asset needs maintenance or not. The CNN-PdM framework presented higher accuracy and lower number of false negatives for both case studies, showing its feasibility to be used as an advisor in the PdM domain.

- There was a lack of research using CNN with electrical data. In this work, different electrical features, such as **power factor**, **energy consumption**, and **amps** are used. The CNN performed well using electrical data; the fact that the CNN-PdM framework outperformed the other approaches testify in favor of the usage of CNN with electrical data as well.

- A novel data transformation method is also proposed: the multiplication method. The usage of the proposed method was crucial to achieving higher accuracy rates, as one can observe in the case studies performed in this work. The transformation through multiplication presented two benefits if compared with GAF: lighter and more heterogeneous images. An example of the difference between multiplication and GAF is here presented: given two features $energy\_comumption - avg = 0.1$ and $amps - sd = 0.2$, for instance, transformed using GAF outputs -0.9548, while transformed using the multiplication method outputs 0.02. The CNN was able to perform better on data transformed by the multiplication technique than on data processed by GAF.

- There was a lack of research on CNN in the PdM domain. In this work, the CNN-PdM framework is proposed to fill this gap. To represent data as an image was a key-factor to benefit most from the CNN's feature representation capability. The CNN was able

to create feature representations that helped in the classification of normal and "faulty" data, being able to outperform the other methods and present a superior accuracy rate.

By choosing predictive maintenance over preventive maintenance, companies can avoid spending a significant part of their budget on equipment that is still in a healthy state, preventing unnecessary costs or downtime of their asset.

5.2 Future Work

This research focused on the possibility of advising building managers about their assets' maintenance schedule. Possible areas for future work would be:

- Since the amount of available data grows with time, deeper CNNs can be generated, which means that higher level of features representations can be achieved. If this is possible, the model's accuracy can be increased.

- If real faulty data become available, it would be possible to predict the remaining time before the next failure occurs. In order to perform this kind of predictions, lots of faulty information must be available. Each maintenance information should be stored in a database to allow the maintenance advisor (PdM system) to calculate how many cycles remaining before each one of the failures. Only with that kind of information, the system would be able to perform predictions based on the current asset's health status regarding the number of cycles remaining until the next failure occurs.

- Another possible problem to tackle would be to predict the asset's component that is going to fail. To allow the PdM system to perform this type of prediction, more specific data regarding the fault type would be needed. Information regarding particular components that were replaced in the maintenance would be required, so the PdM system would be able to learn from it; given the asset's health status, the PdM system would provide information regarding which asset's component is going to fail next time. This

prediction would be helpful to provide insights related to what parts an asset manager should purchase in advance and to better prepare the maintenance team to take action.

Bibliography

[1] L. Xiao, S. Song, X. Chen, and D. W. Coit, "Joint optimization of production scheduling and machine group preventive maintenance," *Reliability Engineering and System Safety*, vol. 146, pp. 68–78, 2016.

[2] FCM, "Federation of Canadian Municipalities." , https://fcm.ca/home/programs/municipal-asset-management-program/municipal-asset-management-program.htm, 2018. Accessed in 26/11/2018.

[3] EFAMA, "European Fund and Asset Management Association." https://www.efama.org/Publications/Statistics/Asset Management Report/EFAMA_Asset Management Report 2018 voor web.pdf, 2018. Accessed in 2018-11-26.

[4] V. Frolov, L. Ma, Y. Sun, and W. Bandara, *Identifying Core Functions of Asset Management*, pp. 19–30. London: Springer London, 2010.

[5] J. Amadi-Echendu, "Managing Physical Assets is a Paradigm Shift from Maintenance," in *IEEE International Engineering Management*, (Singapore), pp. 1156–1160, 2004.

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

[7] L. Yang, X. Ma, R. Peng, Q. Zhai, and Y. Zhao, "A preventive maintenance policy based on dependent two-stage deterioration and external shocks," *Reliability Engineering and System Safety*, vol. 160, no. May 2016, pp. 201–211, 2017.

[8] G. A. Susto, A. Schirru, and S. Pampuri, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, 2015.

[9] S. Selcuk, "Predictive maintenance, its implementation and latest trends," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 9, pp. 1670–1679, 2017.

[10] C. C. Lin and H. Y. Tseng, "A neural network application for reliability modelling and condition-based predictive maintenance," *International Journal of Advanced Manufacturing Technology*, vol. 25, no. 1-2, pp. 174–179, 2005.

[11] T. Borgi, A. Hidri, B. Neef, and M. S. Naceur, "Data Analytics for Predictive Maintenance of Industrial Robots," in *International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, (Hammamet, Tunisia), pp. 412–417, 2017.

[12] J. Ã. Bujak, "Mathematical modelling of a steam boiler room to research thermal efficiency," *Energy*, vol. 33, no. 12, pp. 1779–1787, 2008.

[13] X. Zhang, Q. Chen, R. Bradford, V. Shari, and J. Swithenbank, "Experimental investigation and mathematical modelling of wood combustion in a moving grate boiler," *Fuel Processing Technology journal*, vol. 91, no. 11, pp. 1491–1499, 2010.

[14] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, Curran Associates, Inc., 2012.

[16] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA, USA: PWS Publishing Co., 1996.

[17] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[20] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[21] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] J. Patterson and A. Gibson, *Deep Learning: A practitioner's approach*. O'Reilly, 1 ed., 2017.

[23] F. Chollet, *Deep Learning With Python*, vol. 1. Shelter Island: Manning, 1 ed., 2018.

[24] J. B. Tenenbaum, V. D. Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," vol. 290, no. 5500, pp. 2319–2324, 2000.

[25] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2327, 2000.

[26] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," *CoRR*, vol. abs/1506.00327, 2015.

[27] J.-f. Chen, W.-l. Chen, and C.-p. Huang, "Financial Time-series Data Analysis using Deep Convolutional Neural Networks," *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pp. 87–92, 2016.

[28] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[29] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R news*, vol. 2, no. December, pp. 18–22, 2002.

[30] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak, "An ensemble learning framework for anomaly detection in building energy consumption," *Energy and Buildings*, vol. 144, pp. 191–206, 2017.

[31] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[32] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[33] S. Seo and S.-B. Cho, "Offensive Sentence Classification Using Character-Level CNN and Transfer Learning with Fake Sentences," in *Neural Information Processing* (D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, eds.), (Cham), pp. 532–539, Springer International Publishing, 2017.

[34] H. Muckenhirn, D. Lausanne, and M. Magimai-doss, "End-to-End Convolutional Neural Network-based Voice Presentation Attack Detection," in *IEEE International Joint Conference on Biometrics (IJCB)*, (Denver, CO, USA), pp. 335–341, 2017.

[35] N. Cauchi, K. Macek, and A. Abate, "Model-based predictive maintenance in building automation systems with user discomfort," *Energy*, vol. 138, pp. 306–315, 2017.

[36] M. C. Olde Keizer, R. H. Teunter, and J. Veldman, "Joint condition-based maintenance and inventory optimization for systems with multiple components," *European Journal of Operational Research*, vol. 257, no. 1, pp. 209–222, 2017.

[37] A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari, "Machine learning approaches for improving condition-based maintenance of naval propulsion plants," *Proceedings of the Institution of Mechanical Engineers Part M: Journal of Engineering for the Maritime Environment*, vol. 230, no. 1, pp. 136–153, 2016.

[38] M. Ma, C. Sun, and X. Chen, "Discriminative Deep Belief Networks with Ant Colony Optimization for Health Status Assessment of Machine," *Transactions on Instrumentation and Measurement*, vol. 66, no. 12, pp. 3225–3125, 2017.

[39] K. Leahy, R. L. Hu, I. C. Konstantakopoulos, C. J. Spanos, and A. M. Agogino, "Diagnosing wind turbine faults using machine learning techniques applied to operational data," in *International Conference on Prognostics and Health Management*, (Ottawa, ON, Canada), IEEE, 2016.

[40] M. De Benedetti, F. Leonardi, F. Messina, C. Santoro, and A. Vasilakos, "Anomaly detection and predictive maintenance for photovoltaic systems," *Neurocomputing*, vol. 0, pp. 1–10, 2018.

[41] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. Nascimento, H. Prendinger, and E. M. Henriques, "Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling," *Computers and Industrial Engineering*, vol. 115, pp. 41–53, 2017.

[42] K. Kulkarni, U. Devi, A. Singhee, J. Hazra, and P. Rao, "Predictive Maintenance for Supermarket Refrigeration Systems Using Only Case Temperature Data," in *Annual American Control Conference (ACC)*, (Milwaukee, WI, USA), pp. 4640–4645, 2018.

[43] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine Learning approach for Predictive Maintenance in Industry 4.0," in *14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, MESA 2018*, (Oulu, Finland), pp. 1–6, IEEE, 2018.

[44] H. C. D. G. of Canada, "Historical Climate Data." , http://climate.weather.gc.ca, 2019. Accessed in 24/01/2019.

[45] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: methods and prospects," *Big Data Analytics*, vol. 1, no. 1, p. 9, 2016.

[46] N. L. Tasfi, W. A. Higashino, K. Grolinger, and M. A. Capretz, "Deep neural networks with confidence sampling for electrical anomaly detection," in *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017*, (Exeter, UK), pp. 1038–1045, IEEE, 2018.

[47] D. A. Cieslak and N. V. Chawla, "Learning Decision Trees for Unbalanced Data," in *Machine Learning and Knowledge Discovery in Databases* (W. Daelemans, B. Goethals, and K. Morik, eds.), (Berlin, Heidelberg), pp. 241–256, Springer Berlin Heidelberg, 2008.

[48] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, (New York, NY, USA), pp. 7–10, ACM, 2016.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[50] J. Howard, "Fast AI." , https://course.fast.ai/videos/?lesson=2, 2019. Accessed in 10/02/2019.

[51] J. C.Watkins, *An Introduction to the Science of Statistics : Preliminary Edition*. 2017.

[52] E. W. Weisstein, "Bonferroni Correction."

Appendix A

Data information

This chapter present all the 50 possible features. The features are listed below:

- amps-avgall
- amps-max
- amps-min
- amps-sd
- amps-sum
- apparent power-avgall
- apparent power-max
- apparent power-min
- apparent power-sd
- apparent power-sum
- energy consumption-avgall
- energy consumption-max

- energy consumption-min
- energy consumption-sd
- energy consumption-sum
- hour_cos
- hour_sin
- impedance-avgall
- impedance-max
- impedance-min
- impedance-sd
- impedance-sum
- is_weekend
- month_cos
- month_sin
- outside humidity-avgall
- outside humidity-max
- outside humidity-min
- outside humidity-sd
- outside humidity-sum
- outside temperature-avgall
- outside temperature-max

- outside temperature-min
- outside temperature-sd
- outside temperature-sum
- power factor-avgall
- power factor-max
- power factor-min
- power factor-sd
- power factor-sum
- reactivepower-avgall
- reactivepower-max
- reactivepower-min
- reactivepower-sd
- reactivepower-sum
- real power-avgall
- real power-max
- real power-min
- real power-sd
- real power-sum

Curriculum Vitae

Name:	Willamos Silva
Post-Secondary Education and Degrees	The University of Western Ontario London, ON 2017 - 2019 MESc University of Pernambuco Recife, PE - Brazil 2014-2015 MSc Systems Engineering University of Pernambuco Recife, PE - Brazil 2009-2012 BSc Computer Engineering
Related Work Experience:	Teaching Assistant The University of Western Ontario 2017 - 2019 Senior Software Engineer Accenture (Brazil) 2016 - 2017 Systems Development Analyst Serttel (Brazil) 2015 Junior Software Development Specialist FITec (Brazil) 2012 - 2013