DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
*STOCKHOLM, SWEDEN 2022*

# Predictive Maintenance of Induction Motors using Deep Learning

## Anomaly Detection using an Autoencoder Neural Network and Fault Classification using a Convolutional Neural Network

Diego Andrés Moreno Salinas

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

## Authors

Diego Andrés Moreno Salinas <dmoreno@kth.se>
Information and Communication Technology
KTH Royal Institute of Technology

## Place for Project

Stockholm, Sweden

## Examiner

Dr. Mark Smith
Stockholm, Sweden
KTH Royal Institute of Technology

## Supervisor

Dr. Masoumeh (Azin) Ebrahimi
Stockholm, Sweden
KTH Royal Institute of Technology

# Abstract

With the fast evolution of the Industry 4.0, the increased use of sensors and the rapid development of the Internet of Things (IoT), and the adoption of artificial intelligence methods, smart factories can automate their processes to vastly improve their efficiency and production quality. However, even the most well cared-for machines develop faults eventually. Given that Prognostics and Health Management (PHM) is an indispensable aspect for proper machine performance, Predictive Maintenance (PdM) is an emerging topic within maintenance methodologies whose aim is to predict failure prior to occurrence with the goal of scheduling maintenance only when needed.

As data can be collected faster than ever before, deep learning is an effective tool that can leverage big data for data-driven fault diagnosis methodologies. This thesis explores two different fault diagnosis methodologies associated with predictive maintenance: an anomaly detection using an Autoencoder Neural Network, and a fault classifier using a Convolutional Neural Network (CNN). The system under analysis is a 3-phase AC induction motor commonly used in industry. Results show great performance and indicate the viability for the implementation of both methods in production applications.

## Keywords

Predictive maintenance, Fault Diagnosis, Anomaly Detection, Deep Learning, Machine Learning, Induction Motors

# Abstract

Med den snabba utvecklingen av industri 4.0, den ökade användningen av sensorer och den snabba utvecklingen av *Internet of Things* samt införandet av metoder för artificiell intelligens kan smarta fabriker automatisera sina processer för att avsevärt förbättra effektiviteten och produktionskvaliteten. Även de mest välskötta maskinerna utvecklar dock fel så småningom. PHM är en oumbärlig aspekt för korrekt maskinunderhåll. PdM är ett nytt ämne inom underhållsmetodik som syftar till att förutsäga fel innan de inträffar, med målet att planera underhållet endast när det behövs.

Eftersom data kan samlas in snabbare än någonsin tidigare är djupinlärning ett effektivt verktyg som kan utnyttja stora datamängder för datadrivna metoder för feldiagnostik. I den här uppsatsen undersöks två olika metoder för feldiagnostik i samband med förebyggande underhåll: en anomalidetektion med hjälp av ett neuralt nätverk med autoencoder och en felklassificering med hjälp av ett CNN. Det system som analyseras är en induktionsmotor med 3-fas växelström som ofta används inom industrin. Resultaten visar på goda resultat och visar att det är möjligt att genomföra båda metoderna i produktionstillämpningar.

## Nyckelord

Prediktivt underhåll, feldiagnostik, upptäckt av avvikelser, djup inlärning, maskininlärning, induktionsmotorer

# Acknowledgements

I would like to thank my KTH examiner Dr. Mark Smith and KTH supervisor Dr. Ebrahimi for their continued support and technical advise.

Knightec AB, the sponsor company for this thesis, Mattias Abelsson for giving me the opportunity to work on such thesis, and Enic Amettler, whom with I worked alongside for the whole duration of the project.

The Solna Kolektiv: Johan Broberg, Oscar Svensson and Francesco Ferranti, for making the home-office enjoyable.

Last but not least, my family: Mom, Dad, Poncho, and Marcelo, who have been lifelong supporters of my hopes and aspirations, and have enabled and motivated me throughout my life.

# Acronyms

| | |
|---|---|
| **Adam** | adaptive moment estimation |
| **ARR** | analytical redundancy relations |
| **AUC** | area under the curve |
| **ANN** | Artificial Neural Network |
| **CWRU** | Case Western Reserve University |
| **CIV** | Condition Indicator Value |
| **CAE** | Contractive Autoencoder |
| **CNN** | Convolutional Neural Network |
| **DBN** | Deep Belief Networksk |
| **DL** | Deep Learning |
| **DAE** | Denoising Autoencoder |
| **EMD** | Empirical Mode Decomposition |
| **EEMD** | Ensemble Empirical Mode Decomposition |
| **ENN** | Extension Neural Network |
| **FFT** | Fast Fourier Transform |
| **FTC** | fault tolerant control |
| **HHT** | Hilber Huang Transform |

| | |
|---|---|
| **hp** | horsepower |
| **IoT** | Internet of Things |
| **k-NN** | k-Nearest Neighbors |
| **LSTM** | Long Short-Term Memory |
| **MAE** | mean absolute error |
| **MSE** | mean squared error |
| **ML** | Machine Learning |
| **MCSA** | Motor Current Signature Analysis |
| **PLS** | Partial Least Squares |
| **PdM** | Predictive Maintenance |
| **PvM** | Preventive Maintenance |
| **PCA** | Principal Component Analysis |
| **PHM** | Prognostics and Health Management |
| **ROC** | receiver operating characteristic curve |
| **RUL** | Remaining Useful Life |
| **RPM** | revolutions per minute |
| **SIFT** | scale-invariant feature transform |
| **SVD** | Singular Value Decomposition |
| **SURF** | speeded up robust features |
| **SGD** | Stochastic gradient descent |
| **SVM** | Support Vector Machine |
| **2DNMF** | two-dimensional non-negative matrix factorization |
| **VAE** | Variational Autoencoder |
| **WT** | Wavelet Transformation |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

With the fast evolution of the Fourth Industrial Revolution, or Industry 4.0, the widespread increased use of sensors and the development of massive data collection and information systems (Internet of Things), and the adoption of artificial intelligence methods, smart factories can automate their processes to vastly improve their efficiency and production quality [1].

As manufacturing processes become more complex and throughput increases, equipment downtime and machine failure must be kept to a minimum in order to decrease manufacturing costs and increase plant and personnel safety. Unplanned downtime, however, is an absolute truth of manufacturing. For this reason, PHM is an indispensable aspect of manufacturing responsible of monitoring and analysing the status of an equipment for suitable machine maintenance and proper functionality.

Maintenance methods have rapidly evolved from reactive maintenance, where maintenance is performed after a failure occurs, to Preventive Maintenance (PvM), where maintenance is performed at set time intervals, and into predictive maintenance. Predictive maintenance is a popular topic that has been gaining attention amongst manufacturing companies and academic research, due to its advantages over other maintenance methodologies such as: decrease factory downtime, decrease in maintenance and manufacturing costs, increase in safety, production, equipment life, and an increase in the overall profit, among others [2].

Predictive maintenance aims to detect component and equipment failure before it happens, and ultimately, predict the Remaining Useful Life (RUL) of the equipment, with the goal of allowing manufacturing companies to schedule maintenance before a failure occurs.

As smart factories monitor equipment and collect more data than is possible for technicians to inspect [3], the opportunity for data-driven fault diagnosis methods is opened. Deep Learning (DL) is a subbranch of Machine Learning (ML) that makes use of neural networks with three or more layers which is able to learn from vast amount of data. Because machine learning development rapidly increases and given the vast amount of data that is collected by condition monitoring systems [4], deep learning appears to be a perfect candidate for processing the data and performing fault diagnosis.

The induction motor is of one the most important components in present day manufacturing factories because of its ruggedness, reliability, and cost [5]. It is therefore of utmost importance that proper monitoring and maintenance is performed on induction motors in order to keep manufacturing processes as lean and efficient as possible.

## 1.2 Research Problem and Objectives

The thesis was proposed by Knightec AB, a Swedish consulting company specializing in technology and digital solutions. Knightec's intention was to explore the vast and broad area of predictive maintenance. Given that no project specifications, requirements or metrics were provided, it was up to the author and his project partner, KTH master student Enric Ametller, to define the project. The thesis workload was split into two broad modules: 1) data acquisition and preprocessing, performed by Enric Ametller; and 2) data processing, feature engineering, and model selection and training performed by the author.

Predictive maintenance is a topic of great interest in industry. As explained in Section 2.2, predictive maintenance aims to extend the life of machinery and components by predicting faults and enabling proper scheduling of maintenance. The objective of the thesis is to explore two different deep learning data-driven fault diagnosis approaches for induction motor prognostics. The first one is a semi-supervised anomaly detection

method that uses an autoencoder neural network, while the second one is a supervised fault classification method based on a convolutional neural network. One of the datasets used in the thesis, the Ametller dataset generated by Enric Ametller, will contain data from different types of sensors. From these, the following research questions emerge:

- What kinds of sensor readings (temperature, vibration, pressure, etc.) have the most beneficial impact on predictive maintenance?
- How can deep learning be used for predictive maintenance?

## 1.3 Challenges

One of the biggest problems for a successful predictive maintenance plan is data availability. Machine learning and deep learning algorithms depend heavily on the quantity and quality of the data used to train them. Therefore, without a considerable amount of quality data, the data-driven approach is hard to perform.

In this thesis, fault diagnosis of induction motor data is performed. Upon acquiring such motor, no data existed. This fact was an impediment for selecting and progressing in the thesis. One solution to this problem was to use a public dataset with similar characteristics that can be temporarily used to train and test the models and methodology. The datasets used in this thesis are described in more detail in Section 4.2.

With the explosion of IoT sensor availability, newer equipment incorporate sensors that can be used to collect data. However, as induction motors are very reliable and have an average life of 15-20 years [6], existing factories must find a way to collect data on equipment already on the field. Adding sensors to existing equipment is expensive and, although might be a valuable investment, is not available for everybody. Predictive maintenance and its implementation is a relatively new topic in which the know-how is not readily available.

## 1.4 Delimitations

- The computation of a Condition Indicator Value (CIV) or the RUL will not be explored in this thesis. The computation of a CIV requires motor expert

knowledge for proper selection of health variables while the computation of the RUL requires vast historical data which was not provided by the company and thus not available for the project.

- This thesis focuses on two methods of fault detection and classification from offline data running on a PC. Although the data itself was gathered with an embedded device, the ESP32, the prediction algorithms is are not run real-time on the microcontroller.

## 1.5 Benefits, Ethics and Sustainability

Early machine fault detection and proper maintenance of industrial equipment leads to increased safety, increased machine up-time, longer equipment life and part-life optimization. All of these benefits will ultimately mean better and leaner manufacturing practices with reduced costs that can be transferred directly into the consumer and society. An increase in equipment efficiency also translates into less waste and less resources required.

## 1.6 Outline

The thesis is organized as follows: Chapter 2 introduces all the technical background needed to understand the methodology and implementation of the thesis. Chapter 3 presents literature review compilation of academic works in the area. Chapter 4 describes the datasets used and the methodology implemented. Chapter 5 presents the results and discussions obtained from such experiments. Lastly, Chapter 6 provides concluding arguments along with recommendations for future work.

# Chapter 2

# Theoretical Background

This chapter details the theoretical and technical background needed to understand the proposed data-driven fault diagnosis methods for induction motors. An overview of maintenance methodologies and fault diagnosis methods are presented, followed by technical background on machine learning and deep learning. Lastly, detailed information on autoencoders and convolutional neural networks is presented, as they are the specific methods used for fault detection and fault classification.

## 2.1 Maintenance Methods

Maintenance is an important aspect of manufacturing that increases equipment reliability and quality while decreasing equipment downtime. Maintenance can be grouped into three different categories depending on the methodology used: reactive maintenance, preventive maintenance, and predictive maintenance [1, 7, 8].

- **Reactive Maintenance**, also commonly referred to as Run 2 Failure (R2F), is the simplest but the most inefficient and costly maintenance method. Reactive maintenance involves in repairing the equipment at hand after a fault has occurred, which may have a negative impact in other components as a consequence of the original fault.
- **Preventive Maintenance (PvM)** constitutes of scheduling maintenance at periodic, fixed-intervals to troubleshoot and replace components in order to maintain the equipment. PvM, however, does not take into account the equipment health or status, which could lead to unnecessary costs and corrective

actions such as early replacement of healthy components producing unnecessary downtime [1].

- **Predictive Maintenance (PdM)** takes into account the equipment's health for scheduling maintenance as soon as the equipment experiences deviation from normal operating conditions [1] and before actual breakdown occurs [7]. In this way, the equipment may receive maintenance as needed, avoiding unnecessary component replacement, and thus, extending the component life and minimizing the machine downtime [8]. Predictive maintenance is made possible by the vast amount of data gathered by sensors, and can be thought of as an extension and evolution to Condition Monitoring (CM). Predictive maintenance is the most efficient type of maintenance and is subject to attention by industry because of its benefits. Predictive maintenance, however, could be difficult to carryout if historical data is nonexistent and expensive to implement without the proper hardware and analysis know-how.

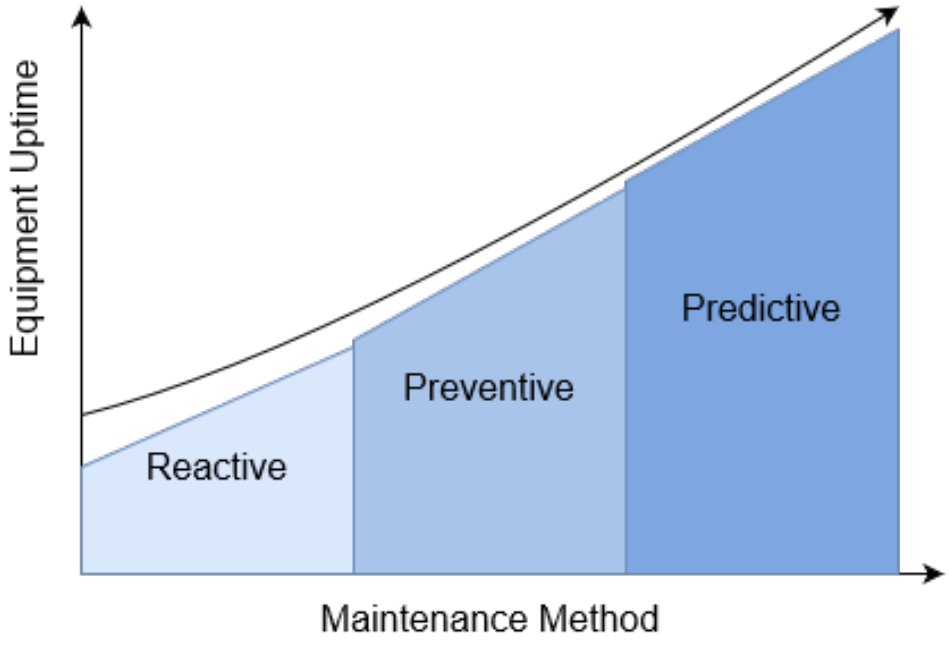



Figure 2.1.1: Maintenance Strategies

## 2.2 Fault Diagnosis

Fault diagnosis techniques that analyze the equipment's operating conditions to detect faults early is a necessary process for improved manufacturing performance [9]. Fault diagnosis methods are commonly divided into three main groups depending

on the techniques and *a priori* information used; model-based methods, signal-based methods, and knowledge-based methods.

- **Model-based** methods use mathematical and physical modelling to mimic the behavior of the equipment under study. Model-based methods require expert knowledge of the equipment itself as well as physical, system and application behavior. Fault diagnosis methods based on model-based methodology monitor the difference between the physical systems' measured outputs and the model's predicted outputs [10].

- **Signal-based** methods utilize signal processing techniques for analyzing the equipment's measured operating condition, highlight and extract features, which would otherwise not be easily visible, and performing a diagnostic decision based on prior knowledge of healthy systems. These methods still require expert knowledge in order to identify faulty signal symptoms. Signal-based methods are usually divided between time-domain (standard deviation, trends, slope and magnitudes), frequency-domain (motor current signature analysis), and time-frequency (Fourier transform, wavelet transforms, Hilbert–Huang transform, instantaneous power FFT, high resolution spectral analysis, wavelet analysis, bispectrum, Park's vector approach, adaptive statistical time-frequency method) signal-based approaches [11], depending on the signal processing method used [10].

- **Knowledge-based methods**, also known as data-driven methods, use large amounts of historical data and can potentially extracts features automatically without any prior signal processing techniques. Unlike model and signal based methods, data-driven methods do not require *a priori* information and thus, allows for analysis to be done without expert knowledge [12]. Data-driven methods commonly utilize statistical and probabilistic analysis, as well as artificial intelligence (machine learning and deep learning) techniques. Data-driven methods are currently experiencing an increase of attention by academia with promising results (Section 3).

## 2.3 Machine Learning

Machine learning is a subbranch of Artificial Intelligence which can be defined as a set of methods used to automatically detect patterns in input data in order to make predictions on future data [13]. Contrary to traditional programming where the designer defines the rules (program) to be used with the data in order to obtain a result, machine learning systems are trained to extract the rules based on the data and results of previous iterations (see Fig. 2.3.1).

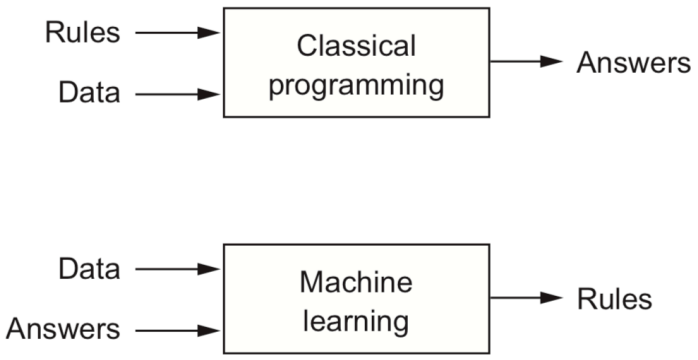



Figure 2.3.1: Machine Learning Paradigm [14]

Machine learning algorithms can be used to solve tasks which are hard for traditional programming to tackle, some of the most common being classification, regression, machine translation, and anomaly detection. Depending on the type of data at hand, machine learning algorithms are broadly categorized as unsupervised learning, supervised learning, and semi-supervised. Unsupervised learning uses unlabeled data that does not have an explicit mapping between inputs $x$ to outputs $y$, such that $D = (x_i)_{i=1}^{N}$. The goal of unsupervised learning is to learn the probability distribution of the data and find patterns in the data. Unsupervised learning is most commonly used to cluster data based on its features in order to discover latent factors intrinsic in the data [13]. On the other hand, supervised learning uses labelled data to learn a mapping from inputs $x$ to outputs $y$ (also called targets), such that $D = (x_i, y_i)_{i=1}^{N}$, where $D$ represents the dataset and $N$ is the number of samples in the dataset [13]. The goal of supervised learning is to learn to predict $y$ from future, unseen, unlabelled data inputs $x$ [15]. Furthermore, supervised learning can be organized into two main types of problems: classification and regression. Classification problems are aimed at learning a mapping from inputs $x$ to outputs $y$, where $y\epsilon\{1, ..., C\}$, with $C$ being the number of discrete classes in a finite set. If the number of classes is equal to two, then

the problem is defined as a binary classification, and if the number of classes is greater than two, the problem is defined as a multi-class classification. Regression problems, like classification problems, are also aimed to learn a mapping from inputs $x$ to outputs $y$, however, $y$ is a continuous, real-valued output $y \epsilon \mathbb{R}$. Lastly, when only a small part of data is labeled, a semi-supervised model may be used instead of having two separate unsupervised and supervised models. Semi-supervised learning uses both unlabeled examples from $D = (x_i)_{i=1}^{N}$ and labeled examples from $D = (x_i, y_i)_{i=1}^{N}$ to predict $y$ from $x$ [15].

The overall goal of machine learning models is to be able to generalize well on new data, also defined as the model's capacity. A model that does not perform well on the training data and is not able to generalize well is said to be underfitting. A model that performs well on the training data but does not generalize well is said to be overfitting. The balance between underfitting and overfitting models is a trade-off between generalization and prediction performance.

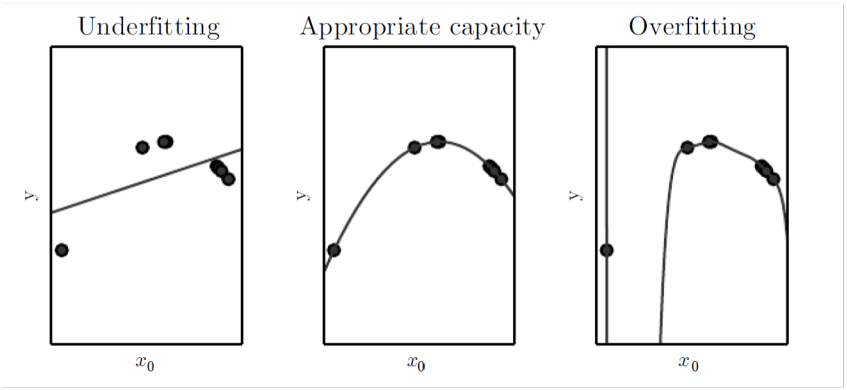



Figure 2.3.2: Model Generalization [15]

In order for machine learning algorithms to generalize and perform well on new, unseen data, the model undergoes a learning process, also known as model training. Models are trained iteratively by comparing the predicted value $\hat{y}$ with the real value $y$, and calculating the prediction error. The models then learn and improve their prediction performance by minimizing such error. The prediction error quantifies the performance of the model, and it is calculated by a loss function. There exist many different loss functions that take different considerations into account to quantify such error (Section 2.4.2.1). The mechanism in which the model is updated such that the prediction error is minimized is called the optimization algorithm (Section 2.4.2.2). The use of different loss functions and optimizers, which themselves have internal

parameters, affect the model's performance. A more detailed explanation is presented in Section 2.4.2.

## 2.4 Neural Networks and Deep Learning

Artificial Neural Network (ANN)s are computational network models inspired on the biological brain composed of connected elements known as neurons. Neurons process information by performing non-linear computations, also known as activation functions (see Section 2.4.1), on its input data, also referred to as features. Features have a weighted value assigned to them that represents their importance.

Deep Learning is a subbranch of Machine Learning and an extension of ANNs that exploits the use of many hidden layers to extract patterns and information from data. Deep Learning focuses on hierarchical representations learning which achieves learning by the use of such successive layers of increasingly meaningful representations [14].

### 2.4.1 Activation Functions

The activation function processes the input data and defines the output value. There are many activation functions commonly used, each of which have different properties and ranges, and are used for different purposes [16]. It is common for all hidden layers to use one type of activation function and the last-layer, or output layer, to use a different activation function depending on the application or prediction required [17]. A hidden layer is one that does not directly interact with the input data and whose output is fed into another hidden layer or the output layer. Artificial Neural Networks may have zero or more hidden layers, while Deep Neural Networks typically have three or more hidden layers [18]. The following is a list of the most common activation functions used:

- The **linear function** is the simplest activation function which only scales the input $x$ by a constant $c$ (2.1) [19].

$$linear = f(x) = c * x \tag{2.1}$$

- The **REctified Linear Unit**, also known as ReLU, truncates negative values

into 0 and has no upper boundary (2.2). ReLU is commonly used since it works well in many applications (insert reference here) [20].

$$ReLU = f(x) = \max(0, x) \tag{2.2}$$

- The **softmax** activation function outputs the respective probabilities of the target labels, all of them adding to 1 [21].

$$softmax = \sigma(x_j) = \frac{e^{x_j}}{\sum_{k=0}^{K} e^{x_k}} \tag{2.3}$$

- The **sigmoid** function is a non-linear function that pushes values towards 0 and 1. The sigmoid function is and effective function for classification purposes [22].

$$sigmoid = f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

- The **tanh** function is similar to the sigmoid function, except that the limits are -1 and 1 [23].

$$tanh = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

### 2.4.2 Network Training

The way in which deep learning models improve their prediction capabilities is by a method called Gradient Descent. As a model's weights and parameters are usually randomly initialized, gradient descent works by updating these parameters with the goal of minimizing the loss function (see Section 2.4.2.1) via an optimization algorithm (see Section 2.4.2.2). The term gradient descent gets its name from the approach the parameter update is performed: in order to minimize the loss function, the weight's derivative with respect to the weight in the previous layer are computed for every layer and consecutively updated. As shown in Figure 2.4.1, the initial weight is gradually updated until the minimum cost is reached, process also known as *backpropagation* [24].

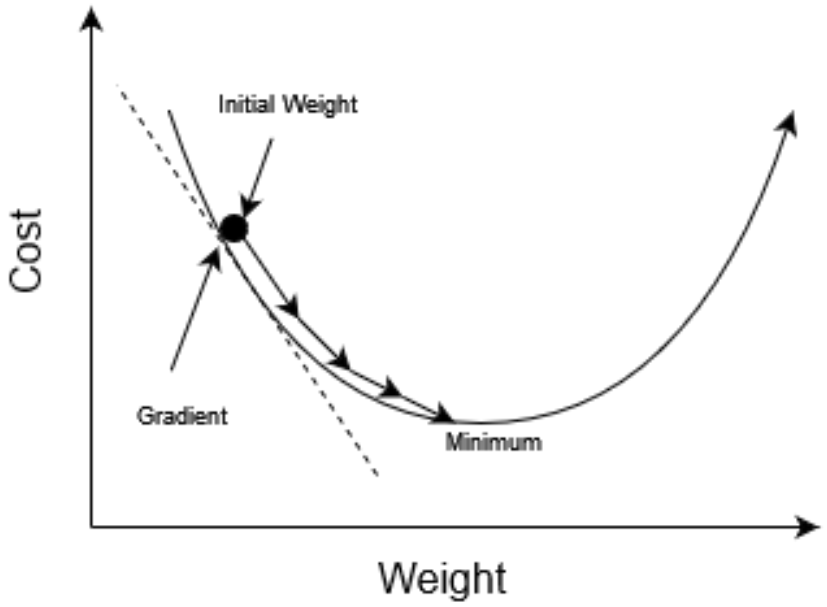



Figure 2.4.1: Gradient Descent

The efficacy of the training process, however, is dependent on different parameters that affect the way the weights are updated, with the most important ones being the learning rate and the batch size. These tunable parameters are also known as the model's *hyperparameters*. The *learning rate* is the size of the step when moving towards the minimum at each iteration when updating the weight values. If the learning rate is set too small, then the convergence to a minimum will be very slow. On the other hand, if the learning rate is set too large, the model might not converge at all [13]. The *batch size*, or the number of training examples used for estimating the derivative of the loss function, also has an impact on the performance model training. Updating the weights after each iteration would result in a more accurate computation but more computationally expensive. Inversely, updating the weights using all of the training samples would be less accurate but far less computationally expensive [14]. The right selection of the optimization method and its hyperparameters are key to proper model training.

The most common way of training and testing a model is to randomly split the dataset into subsets to be used for training, validating, and testing the model. The training set is used during model training and function loss optimization, the validation set can be used in conjunction with the training set to observe the model's performance and hyperparameter optimization, and lastly, the test set is used to evaluate the model after it has been trained. Doing so, we guarantee all the data used have the same

distribution. *K-fold cross-validation* is a methodology for partitioning the dataset into *k* non-overlapping subsets and using different subsets as training, validation, and test sets, and training the model *k* times while using different subsets for different purposes [15].

A brief overview of the most common loss functions and optimizer algorithms used will be presented. For in-depth know-how of how network training is performed, the author refers the reader to [14, 15].

#### 2.4.2.1 Loss Functions

The loss function, also called the objective function, is a function that quantifies how well the model is performing. In general terms, the loss function is the error between the expected value and the current model output. The minimization of the loss function is the goal during training. In other words, the less the error between predicted and actual values, the better performance the model has [24]. Different loss functions quantify the error in different ways, and therefore penalize different predicted results differently. The most commonly loss functions used for regression are the Mean Absolute Error and the Mean Squared Error, while the Categorical Cross-Entropy loss function is commonly used for classification problems.

- The **mean absolute error (MAE)** loss function measures the mean of the absolute difference between the predicted values, $\hat{y}_i$, and the the actual values $y_i$. The MAE loss penalizes deviations from the mean in either direction equally, where the optimal prediction will be the median, and therefore, this loss function is not sensitive to outlier values [25].

$$MAE = L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid \tag{2.6}$$

- The **mean squared error (MSE)** loss function measures the mean of the squared differences between the predicted values, $\hat{y}_i$, and the the actual values $y_i$. The MSE loss penalizes deviations from the mean in either direction equally, however, harshly since it squares all the errors, and therefore, this loss function

is sensitive to outlier values [26].

$$MSE = L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.7}$$

- The **Categorical Cross-Entropy** function is used for multi-class, single-label classification tasks, where the model has to select only one label out of many. The loss function is calculated by computing formula 2.8, where $\hat{y}_i$ is the $i$-th value of the model output and $y_i$ is the actual value [27].

$$CC = L(y, \hat{y}) = \sum_{i=1}^{N} y_i \cdot \log \hat{y}_i \tag{2.8}$$

- The **Binary Cross-Entropy** function is used for multi-label, binary classification tasks, where the model has to select one label out of two options for one or multiple attributes. The difference between binary and categorical cross-entropy is that the former is used when selecting a label out of two options while the latter selects one label out of many [28].

$$BC = L(y, \hat{y}) = -\frac{1}{output\ size}\sum_{i=1}^{output\ size} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log 1 - \hat{y}_i \tag{2.9}$$

The right selection of the last-layer activation is crucial for proper use of the loss function. Table 2.4.1 illustrates the most commonly last-layer activation and loss function combinations used.

Table 2.4.1: Last-layer Activation and Loss Functions [14]

| **Problem Type** | **Last-Layer Activation** | **Loss Function** |
|---|---|---|
| Binary classification | sigmoid | binary_crossentropy |
| Multiclass, single-label classification | softmax | categorical_crossentropy |
| Multiclass, multilabel classification | sigmoid | binary_crossentropy |
| Regression to arbitrary values | None | mse |
| Regression to values between 0 and 1 | sigmoid | mse or binary_crossentropy |

#### 2.4.2.2 Optimization Methods

As explained in 2.4.2, the goal of model training is the minimization of the loss function via iterative updating of the network weights. The lower the error between the real and predicted values, the better the model's performance. There are a few different optimizer algorithms that differ in the way the weights are updated by utilizing different learning rates and batch size paradigms. For example, some methods vary the learning rate in order to fine-tune the learning as the model evolves, making use of *momentum* (using previous weight updates into account) [14] and *decay* (slowly decreasing the learning rate value). The following is a list of the most commonly optimization algorithms used:

- **Stochastic gradient descent (SGD)** is an implementation of gradient descent that utilizes either a single (*true* SGD) or a subset (*mini-batch* SGD) of the samples when performing the update (contrary to gradient descent that utilizes the whole set of samples) [14].

- **Adagrad**, short for Adaptive Gradients, is an extension of gradient descent that automatically modifies the learning rate size to adapt to the gradients [29].

- **RMSProp**, short for Root Mean Squared Propagation, is an extension of gradient descent and a modification of Adagrad that uses a moving average to keep track of the gradients when performing the weight update, thus utilizing most recently seen samples [30].

- The **adaptive moment estimation (Adam)** is a gradient-based optimization objective function, based on adaptive estimates of the first and second moments [31]. Adam is a widely used optimization algorithm because it is computationally efficient, requires little memory, and delivers good performance. Adam combines the advantages of two other well known optimization algorithms; just like the Adaptive Gradient Algorithm (AdaGrad), Adam maintains a per-parameter learning rate; and like Root Mean Square Propagation Algorithm (RMSProp), Adam maintains a per-parameter learning that adapts to the gradient derivatives [32].

### 2.4.3 Evaluation Metrics

After the model has been trained, different evaluation metrics are used to quantify the performance of the model. In other words, evaluation metrics quantify how well the model generalizes to newly, unseen samples [33]. For classification problems, the confusion matrix is a useful table layout that summarizes how well a model's predictions were. As seen in Table 2.4.3, the confusion matrix provides a easy visualization of the true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). A True positive and true negative correspond to having a correct prediction of positive and negative, respectively. A false positive corresponds to incorrectly predicting a positive value while a false negative corresponds to incorrectly predicting a negative value. In a perfect model, the diagonal contains all the predictions and the non-diagonal is null, meaning that the model has correctly predicted all the data samples.

Table 2.4.3: Sample Confusion Matrix

| | | Predicted Values | |
|---|---|---|---|
| | | Positive | Negative |
| Actual Values | Positive | TP | FN |
| | Negative | FP | TN |

- **Accuracy** is the percentage of correct predictions amongst the total, with values between 0 and 1.

$$accuracy = \frac{correct}{total} \tag{2.10}$$

- **Precision** measures the fraction of correctly predicted true positives in the positive class, with values between 0 and 1.

$$precision = \frac{TP}{(TP + FP)} \tag{2.11}$$

- **Recall** measures how well the positive class was identified, with values between 0 and 1.

$$precision = \frac{TP}{(TP + FN)} \tag{2.12}$$

- **F1-Score** is the harmonic mean of the precision and recall scores, with values

between 0 and 1.

$$f1 - score = \frac{2 * precision * recall}{(precision + recall)} \tag{2.13}$$

## 2.5 Autoencoders

Autoencoders are a type of artificial neural networks that can be used to extract compressed representations from data in a self-supervised manner, given that the targets are generated from the input data. An autoencoder is composed of three components: an encoding module, a decoding module, and a loss function. With these three components, the network is trained to learn an approximation to the identity function such that $y = x$, where $x$ is the input layer and $y$ is the output layer. To prevent the network from creating a trivial mapping from input to output, the autoencoder's middle layer $h$ is constrained to be narrow (with less hidden units) [13]. This low dimension representation layer is called the embedding or bottleneck (see Fig. 2.5.1).

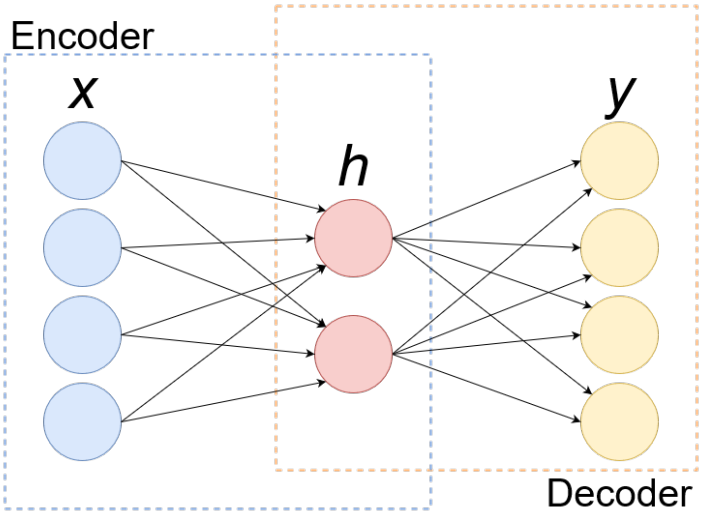



Figure 2.5.1: Autoencoder Structure

The encoder module is usually composed of Fully Connected layers with decreasingly number of hidden units used to map the input data to the lower dimension representation of the data. On the other hand, the decoder module is usually composed of Fully Connected layers of increasingly number of hidden units and is used to map the lower dimension representation into the output data (see Fig. 2.5.1). For predictive maintenance applications, an autoencoder model can be used to learn the

behavior of an equipment during its normal condition; any deviation from such learnt representation might correspond to an anomalous behavior.

## 2.6 Convolutional Neural Networks

Convolutional Neural Networks are a type of artificial neural networks that are commonly used in computer vision and image-classification applications, given that they are particularly good at finding and learning local patterns in image data (pixel values), and even data that has a known, grid-like topology such as time-series data taken at 1-D time intervals [15].

Convolutional Neural Networks have some properties that make them effective in image processing applications: they provide translation invariant pattern recognition, they are able to learn spatial hierarchies of patterns, and they allow for parameter sharing, making them smaller than regular ANNs. Translation invariant means that the patterns learned are independent of their location within an image, while spatial hierarchy learning means that they are able to learn increasingly complex and abstract visual concepts as the information is passed thought the layers [14]. Lastly, the total number of parameters in the network is smaller given that the weights are shared across the image or input data [13]. CNNs are able to provide these advantages over regular ANN because of their use of the convolution and the pooling layers. Convolutional layers use a specialized linear operation called the convolution, which works by using sliding filter windows along the feature map. As such, convolutions are defined by two parameters: the size of extracted data from feature maps (filters), and the depth of the output feature map (number of filters). After the convolution layer, the pooling layer is commonly used, allowing the network to summarize the feature maps by downsampling and thus, lowering the number of parameters in the model.

For example, as seen in Figure 2.6.1, three $3x3$ input filters (input patches) are applied to the input data, resulting in output data of different dimensions. In the case of black and white images, the input is normally 1-channel values representing the grayscale pixel intensity, and in the case of colored images, the input is 3-channel values representing the RGB pixel values. In the case of time-series data, such as sensor data, the input is a 1-channel sensor values.

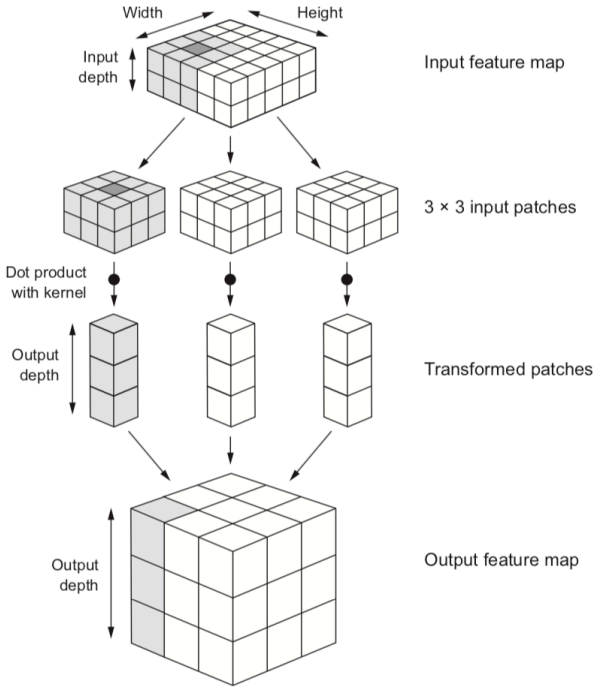



Figure 2.6.1: Convolutional Layer [14]

Many CNN architectures have been developed over the years, each utilizing different number of layers, activating functions, and configurations, and commonly improving upon previous findings. For example, AlexNet, designed by Alex Krizhevsky in 2012, contained five convolutional layers, three pooling layers and three fully connected layers; VGGNet, designed by Karen Simonyan and Andrew Zisserman in 2014, was composed of small convolutional layers that reduced volume size; and LeNet-5, originally proposed by Yann LeCun in 1989, was one of the earliest convolutional neural networks, containing two convolution layers, two pooling layers, and three fully connected layers, and utilizing the tanh activation function, with a total of 60,850 parameters. Figure 2.6.2 shows the architecture of LeNet-5, a simple but widely used convolutional neural network structure [34].

For predictive maintenance applications, and given the powerful classification capabilities of CNN's, Convolutional Neural Networks can be used to train a classifier model that can distinguish between different equipment working conditions.

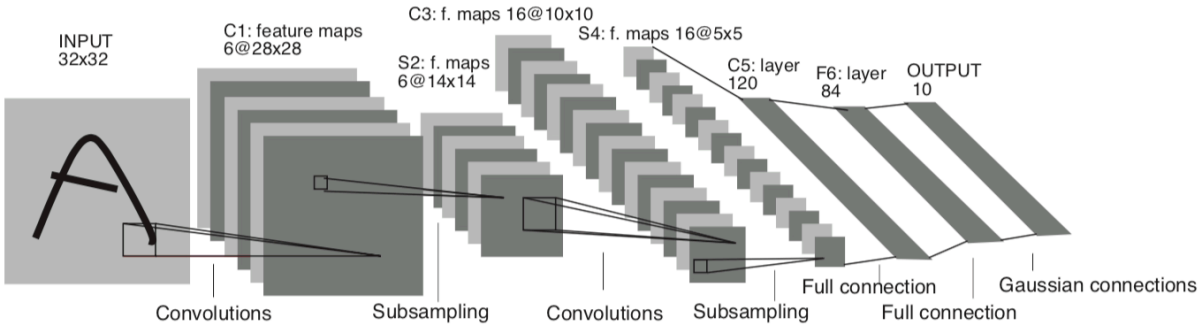



Figure 2.6.2: LeNe-5 CNN Architecture [13]

# Chapter 3

# Literature Review

As described in Section 2.2, fault diagnosis methodologies vary depending on the amount of data at hand. Manufacturing industries rely on motors for their processes with the induction motor being amongst the most important component [9] because of its ruggedness, reliability and cost [5]. Fault detection and diagnosis for moving machinery such as motors and turbines has been a topic of interest by industry for some time now. In past decades, however, the methodologies used have mostly been model-based and signal-based. Given the cost of hardware, the use of well-controlled simulation motor models for testing fault diagnosis approaches is has been the strategy used [35–38]. Such simulations are meant to replicate the motor's dynamic behavior and electromechanical interactions. Simulations, however, require expert knowledge on multiple scientific realms such as magnetism, mechanical, electrical and circuit theory [39] for designing and creating such model. For example, [40] set forth two model-based fault tolerant control (FTC) schemes for wind turbine fault detection using generator torque data, while [41] combines the use of analytical redundancy relations (ARR) and interval observers.

Vibration analysis is frequently used given the mechanical nature of motors. Fast Fourier transform (FFT) [42], Hilber Huang Transform (HHT) [43, 44], Wavelet Transformation (WT) [45], Ensemble Empirical Mode Decomposition (EEMD) and the Empirical Mode Decomposition (EMD) [46, 47] are some of the commonly used signal-processing techniques for feature extraction and vibration analysis. [48] proposed a vibration signal-based fault detection and diagnosis system for induction motors by converting the time-series data into an 2-D image and subsequently perform

the scale-invariant feature transform (SIFT) algorithm to extract significant features. The CWRU dataset (Section 4.2) has been an important vibration sensor benchmark dataset that facilitates the development of algorithms for fault diagnosis purposes. [49] applied three different diagnostic techniques: an envelope analysis, a cepstrum prewhitening, and a combination of discrete/random separation, spectral kurtosis, and envelope analysis.

Current analysis, also known as Motor Current Signature Analysis (MCSA), as the name implies, is another approach based on a motor's stator supply current [5]. MCSA in itself is not a processing method but rather a description of the motor's current behavior under different conditions. Different signal-processing techniques can be utilized for analysing current data; MCSA using Fast Fourier Transform (FFT) [35, 50], instantaneous power [51, 52], demodulated current spectrum [53, 54], wavelet analysis [55], and Park's vector approach [56].

With the rise and advancement of Artificial Intelligence methods, the development of hardware platforms that enable high-throughput parallel computations, a decrease in sensor production cost, and increase in data collected, data-driven methodologies are now of interest and explored by the academic and industrial community. Data-driven methods are those which use large amounts of data for feature diagnosis. Methods can be grouped in to different categories such as statistical, machine learning and deep learning. Statistical analysis methods include Principal Component Analysis (PCA) and Partial Least Squares (PLS) [57]. Machine learning methods are apt for big amounts of data, however, they are usually combined with signal-processing techniques for highlighting the features and further feature extraction (Figure 3.0.1). Common machine learning methods used in motor diagnosis are principal component analysis (PCA) [58], Hilbert–Huang Transform and Support Vector Machine (SVM) [43], Hilbert–Huang Transform with k-Nearest Neighbors (k-NN) [44], Singular Value Decomposition (SVD) [59]. [8] uses Random Forests for analysing the health of a woodworking cutting machine spindle health; [60] uses a multiple classifiers with different prediction horizons approach. [61] use SVM for evaluating different feature extract methods, [62] proposed a Fuzzy Logic based resilient state-awareness of control system for anomalous behavior detection, and [63] used wavelet packet decomposition on sound data along with an Extension Neural Network (ENN) for fault diagnosis in an internal combustion engine.

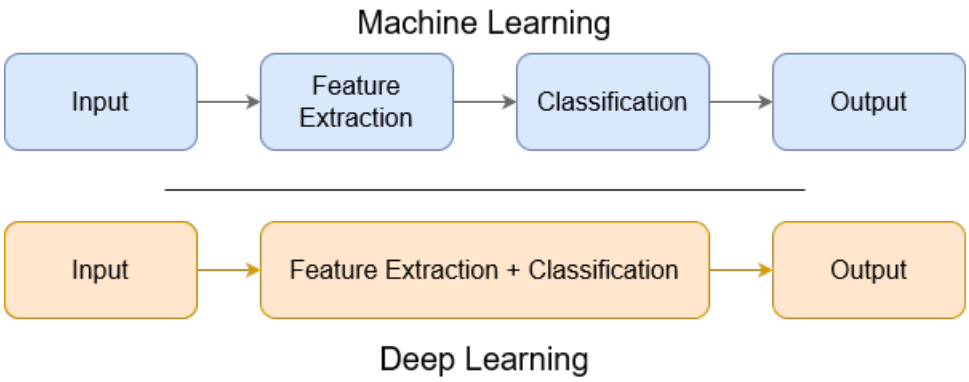



Figure 3.0.1: ML vs. DL Algorithm

Deep learning methods can be treated as black boxes where the feature extraction and the prediction algorithm is combined (Figure 3.0.1). This characteristic makes them highly desirable and customizable to different applications. Within deep learning there are different kinds of network architectures, such as Long Short-Term Memory (LSTM), Deep Belief Networksk (DBN) [64], autoencoders, and CNNs, that are appropriate for different applications and data types. For example, [37] used three-phase current data and the electrical angle as inputs to a LSTM. CNNs have had the most advancement because of its practical applications in image recognition. The use of convolutional neural networks for fault detection applications has been in development for some years now. [65] used generator torque data for CNN fault detection in wind turbines based on [66] benchmark model, [4] used the CWRU dataset for fault diagnosis, [67] and [68] both used a 1-D CNN for real-time motor fault diagnosis and real-time damage detection, respectively, while [69] used a CNN with an adaptive learning rate for fault diagnosis with the CWRU dataset. Other authors have used their own datasets for fault detection and classification purposes; used an adaptive 1-D CNN [67] on motor current data and [9] on vibration data.

When the amount of historical data is small or non-existent, supervised data-driven methods are hard to implement. Unsupervised learning can be used to learn the data's representation of latent features for clustering and classification purposes. Autoencoder neural networks are a good option given their architectural simplicity and their ability to compress and extract features from raw data [70]. [71] used a three-layer sparse autoencoder with the maximum mean discrepancy term for minimizing the differences between the training and test data. In addition to the basic autoencoder [72], different variations such as the stacked autoencoder [73], the stacked Denoising

Autoencoder (DAE) [70, 72], the Contractive Autoencoder (CAE) [72], a kernel-based autoencoder [74], sparse auto-encoder [64, 71], Variational Autoencoder (VAE) [75] have been used for fault diagnosis.

Another alternative is to use semi-supervised learning, where unsupervised learning is initially used to identify the data's distribution and supervised learning is applied for further classification. [76] explored a semi-supervised model that combined PCA analysis for cold start learning and subsequently applied SVN for newly detected faults. [75] used a Variational autoencoder based deep generative model for effective use of a small subset of labeled data.

Hybrid approaches, where a combination of signal-based models and data-driven models, is also frequently used. [7] designed a deep neural network for fault detection and diagnosis of induction motor current by processing the signals with the Hilber-Huang-Transform for EMD, features which where used as inputs to the ANN. In addition, to combat the problem of data imbalance, the authors developed a Generative Adversarial Network to over sample data.

Since sensor data is time-series format, different methods for conversion from 1-D time-series data to 2-D images have been implemented. [48] employs a normalization scheme for translating vibration values into pixel intensity values, [77] utilizes wavelet coefficients deduced from Shannon wavelets, [78] utilizes a generalized S transform and two-dimensional non-negative matrix factorization (2DNMF), [79] uses a bi-spectrum contour map for transformation of signals to images. These methods, however, only concern the data format conversion and not the feature extraction step and thus still require expert knowledge for the feature extraction methods. [48] applied a SIFT, [77] uses multi-class SVM, and [79] uses the speeded up robust features (SURF) feature extraction method. The methodology used in this thesis follows [48], also adopted by [4, 65], in conjunction with deep neural networks for automatic feature extraction.

# Chapter 4

# Methodology

This chapter details the methodology used for the proposed fault diagnosis solutions for anomaly detection and fault detection explored in this thesis: the semi-supervised Anomaly Detection method using an Autoencoder Neural Network and the supervised Fault Classifier method using a Convolutional Neural Network. Section 4.1 gives an overview of the overall purpose, objective and strategy behind the proposed solutions. Section 4.2 provides an overview of the two datasets used for this thesis project. Sections 4.3 and 4.4 describe, for each method, the data processing step, the model architecture, and the workflow followed.

## 4.1 Overview of Proposed Method

The purpose of using an Autoencoder in this thesis is to create a lightweight deep learning model that is be able to detect anomalies and be small enough to fit in an embedded device, given that an autoencoder model could be as simple as a 3 layer neural network. Additionally, in real-life scenarios, many manufacturing companies do not have historical data that can be used to train fault diagnosis models. However, it is possible to retrofit and collect data on existing machinery that is running properly, or in a "normal" state, to be used as a baseline [80]. Any deviation from this baseline will be considered an anomaly. In this thesis, the autoencoder method could be considered to be a "supervised" problem given that we have a labeled dataset to test the algorithm. One downside with this method is that the algorithm only distinguishes between normal and anomalous condition, and does not categorize or classify the origins of such faulty condition. For this reason, the fault classifier with a Convolutional Neural

Network was explored as well. Given a labelled dataset, the classifier is able to pinpoint the root cause of the fault. For each of these methods, vibration, current, temperature data was used for sensor comparison, as well as a unified model that utilizes all data as an input. Finally, the best performing models are compared in term of size, bandwidth and performance. This would work as a proof of concept to test how such models could run on an embedded device.

## 4.2 Datasets

Since this thesis was performed in parallel to Enric Ametller's thesis, the Ametller Dataset was not ready until the end of the semester. Therefore, it was necessary to begin working with another dataset for testing the methodologies used in this thesis:

- CWRU Dataset (Section 4.2.1): used to build the machine learning environment and workflow, and to start testing the deep learning models and architectures.
- Ametller Dataset (Section 4.2.2): used to test the fault diagnosis on a specific induction motor.

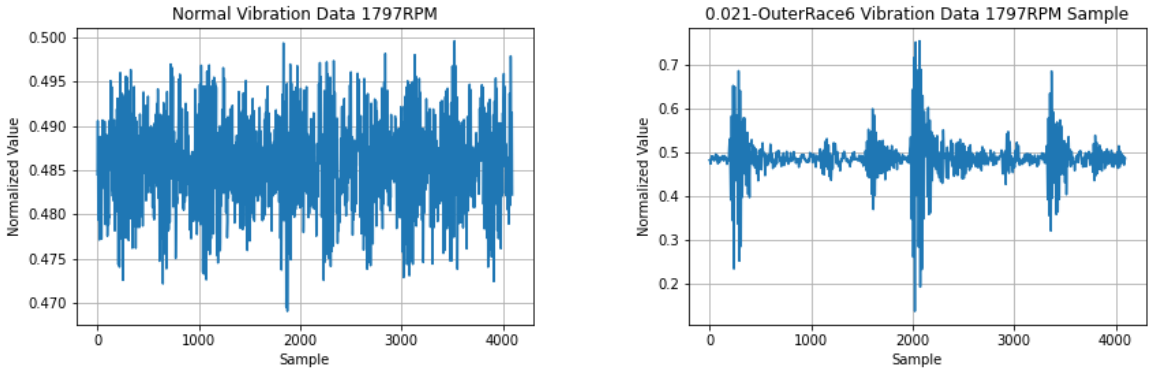
### 4.2.1 CWRU Dataset

The Case Western Reserve University (CWRU) Bearing Dataset [81] is one of the most commonly used datasets for bearing fault detection and fault classification. The dataset was created to characterize the performance of a motor bearing condition assessment system developed at Rockwell, and has since evolved as a benchmark dataset used by numerous research papers [4, 49, 70, 71, 82–88] for testing signal-processing techniques, feature engineering, and data-driven models.

The test bench consists of a 2 horsepower (hp) Reliance Electric motor (left), a torque transducer/encoder for setting a constant load (center), and a dynamometer (right) for measuring the motor's torque and RPM. The dataset consists of vibration data collected at 12kHz and 48kHz with accelerometers placed at the drive-end and the fan-end bearing housings [89]. The bearings were purposely damaged using electro-discharge machining with faults diameters of 7, 14, 21, 28, and 40 mils (0.18, 0.36, 0.54, 0.72, and 1.02 mm respectively) on 3 different locations of the bearings, and thus, 3 fault types: inner race, outer race and ball. SKF bearings and NTN equivalent

bearings were used. Additionally, the experiments were performed at 3 different hp's or revolutions per minute (RPM)'s.

As the outer race faults are stationary faults, the accelerometer sensor was placed in 3 different positions, at the 12, 6, and 3 o'clock. For purposes of this thesis, the Drive End vibration data was utilized. Figure 4.2.1 can be used to visualize the vibration data as provided by the dataset.




(a) Normal (b) Anomalous

Figure 4.2.1: Raw Vibration Data Samples

### 4.2.2 Ametller Dataset

The second dataset utilized in this thesis was developed by Enric Ametller at Knightec AB, a Swedish consulting company.

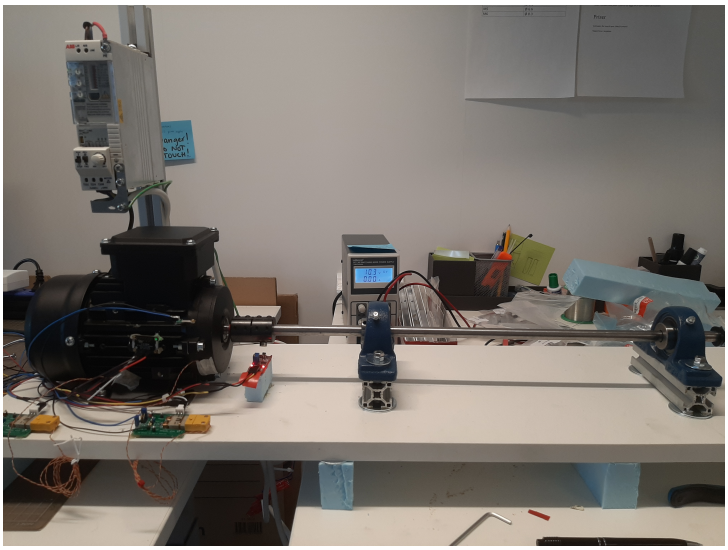


Figure 4.2.2: Ametller Test Rig

The test bench, seen in Fig. 4.2.2, consists of a Busck three-phase, 230VAC, 180W

asynchronous induction motor (left) [90], an ABB AC motor drive [91] used for adjustable speed (top left), and the test bearings (right). The hardware sensor setup is made up of 2 temperature sensors, 1 piezoelectric vibration sensor, 1 3-axis accelerometer vibration data, and 3 current sensors, one for each electrical phase. The data was collected at 5kHz for the 3-axis accelerometer sensor and at 10kHz for the rest of the sensors.

The dataset is composed of 100 files per RPM and per fault. Data was acquired with different experimental setups, or fault condition states, being the following: "healthy", "bearing fault", "high speed", "shaft misalignment", and "bearing+shaft" (both bearing and shaft misalignment faults at the same time). For the "healthy" condition, the data was first acquired under normal/healthy motor condition using brand new SKF bearings. For the "bearing fault", the bearings were subsequently intentionally damaged with 3.0 mm drilling hole on the side race, in order to introduce and simulate a fault. It must be noted that such drilling procedure also introduced dust and impurities to the ball bearings, further damaging the bearings. The AC motor drive utilized provides a variable rotational encoder for controlling the motor speed, along with the option to configure the maximum frequency the motor will reach when running in full speed. Although not a fault condition in itself, running the motor for prolonged periods in such state can cause damage due to temperature and vibration increase. For the "high-speed fault", data was collected under this condition. Shaft misalignment can be generated by improper installation of the motor or mechanical wear of the components. For the "shaft misalignment" fault condition, the load bearings and coupling were purposely and manually misaligned, creating an angle between the motor shaft and the load. Lastly, the "bearing and shaft misalignment" fault combined the experimental methodology of both faults (using a faulty bearing with a misaligned shaft and coupling).

### 4.2.3 Data Preprocessing

Raw data can be found in many different formats and ranges, which can affect the model's interpretation of the data and affect its performance. It is therefore needed to process the data before feeding it into our models. One of the most important steps in the preprocessing pipeline is that of feature scaling.

#### 4.2.3.1 Feature Scaling

Feature scaling is an important step in the data preprocessing pipeline. Raw data values can have different ranges which may cause the model to give more importance to the higher value. As a rule of thumb, it is recommended for features to have *small values*, typically between the 0-1 range, and be *homogeneous*, meaning all features be in the same range [14]. Using large values can create large gradient updates that can cause slow convergence [92] or even prevent the network from converging at all [14]. There are many different feature scaling methods, with the following being the most common:

- **Min-Max Normalization** delimits the values between 0-1. This type of scaling is useful in image processing applications where the pixel intensity needs to be between [0-255] or [0-1] [93]. For each data sample $x$, we subtract the minimum and divide it by the range.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

- **Standardization (Z-Score)** transforms the data to have a zero-mean and a unit-variance. For each data sample $x$, we subtract the mean and divide it by the standard deviation, where $\mu$ is the mean and $\sigma$ is the standard deviation of the feature.

$$x' = \frac{x - \mu}{\sigma} \tag{4.2}$$

- **Mean Normalization** subtracts the mean and divides by the range for each for each of the data samples.

$$x' = \frac{x - \mu}{max(x) - min(x)} \tag{4.3}$$

#### 4.2.3.2 Imbalanced Data

A dataset with skewed class proportions, where one class has significantly more samples than other, is referred to as an imbalanced dataset [94]. An imbalanced dataset can compromise the performance and learning of an algorithm, where the model fails to accurately represent the true distribution of the data, and thus, leading to poor performance [95]. Some of the most common methods for addressing imbalanced

data are the following:

- **Downsampling** corresponds to using less samples from the majority class in order to achieve a balanced distribution [96].

- **Upweighting** involves assigning class weights relative to the class size. In other words, these class weights, which are in turn used by the cost function, aim to "assign a high cost to the misclassification of the minority class", thus minimizing the total cost [96]

- **Use different metrics**: when faced with an imbalanced dataset, using accuracy as a metric may not always be the best idea. The "Accuracy Paradox" is a phenomenon that explains how a high accuracy is not representative of the problem, caused by the majority class masking the obtained results [97]. For example, suppose a dataset that contains a majority class of 99 elements and a minority class of 1 element; predicting all 100 samples as majority class will result in an accuracy of 99%, misclassifying the 1 minority sample we were interested in detecting. For this reason, other metrics such as recall, precision, and F1-score (described in Section 2.4.3) should be utilized.

- **Collect more data**: self-explanatory, this solution calls for collecting more data in order to balance the class distribution. It may not always be possible to do so.

### 4.2.4 Time Series to 2-D Image Conversion

Given the time-series nature of the sensor data, an extra preprocessing step is needed in order to use the data as input to a CNN. For each data file, and depending on the number of data points in a specific file, the conversion algorithm slices the data into segments and stacks them to create $n$ 2-D images of size $M \times M$ (Fig. 4.2.3), resulting in grayscale images, or 1-channel images, with a final size of $M \times M \times 1$, where $M$ is a multiple of $2^n = 16, 32, 64, 128$. In this thesis, $M = 16 - 64$ was mostly used. This conversion methodology is a combination of the signal-to image conversion method used by [4] and [65]. With this time series to image conversion, there exists a column-wise relationship between pixels, and a shifted row-wise relationship. One of the advantages of this method is that no manual feature extraction is required.

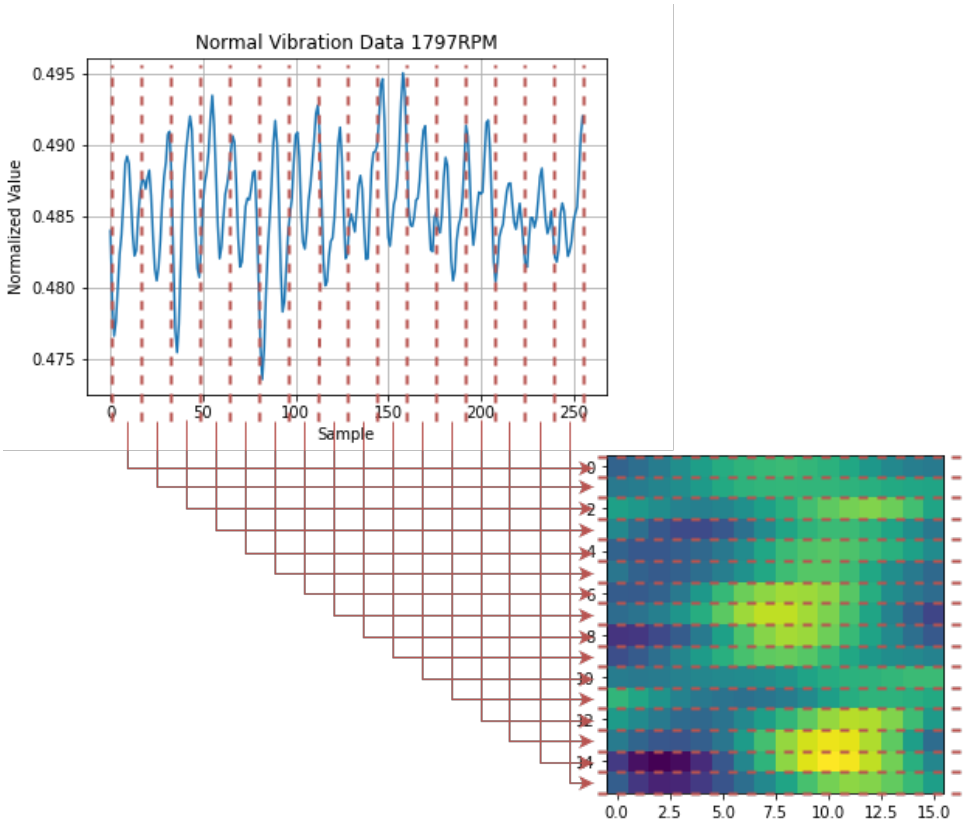



Figure 4.2.3: Conversion from Time Series to a 2-D Image

## 4.3 Anomaly Detection with an Autoencoder

The purpose of an Autoencoder Neural Network is to have the network learn the latent space representation of what the normal condition and proper functionality of the data is by training the autoencoder using only the normal data. By learning the most salient features of the normal data, the autoencoder is able to precisely decode and recreate the input signal. The error between the input signal and the reconstructed output signal is called the *reconstruction error*. The idea is that, when using faulty data, the model should return a high reconstruction error. By setting a threshold or boundary on the reconstruction error, we can distinguish between normal and anomalous data. This proposed method was inspired by [98].

### 4.3.1 Model Architecture

The base model architecture structure of the Autoencoder Neural Network is the same for all experiments: the encoder module, composed of Fully-Connected layers with decreasingly number of hidden units; the embedding layer, the smallest layer; and

the decoder module, composed of Fully-Connected layers with increasingly number of hidden units. Comprehensive experiments to define the size of the data samples, the number of layers in the encoder/decoder modules, the hidden units in each layer, the learning rate, and the number of epochs were performed with a mix of manual tuning and automated tuning with the aid of Keras Tuner, a Tensorflow library that assists with hyperparameter optimization, process also known as *hypertuning* [99].

### 4.3.2 Loss, Optimizer and Evaluation Metrics

The loss function used during compilation was the MAE loss function. As explained in Section 2.4.2.1, for each data point in a sample, the MAE loss [25] is calculated by averaging the absolute values of the reconstruction errors, in other words, the difference between the predicted values, $\hat{y}_i$, and the the actual values $y_i$.

$$MAE = L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \mid y_i - \hat{y}_i \mid \tag{4.4}$$

The Adam optimizer was selected for the optimization algorithm. The reconstructed samples' labels are then predicted depending on the reconstruction error threshold, and thus, selection of proper threshold has great effect on the model's overall performance. For this reason, the metrics used to quantify the performance are the accuracy, precision and recall, values used to create the receiver operating characteristic curve (ROC) and area under the curve (AUC) metrics.

## 4.4 Fault Classifier with a CNN

The purpose of using a Convolutional Neural Network is to have the network classify the type fault occurring in the system under observation. Given a labeled dataset, as is the case in this thesis, the CNN can act as a fault classifier to indicate the cause of failure. After processing the time-series data and converting it into images, as detailed in Section 4.2.4, the images are used as input for training the CNN model.

### 4.4.1 Model Selection

A model based on LeNet-5 [4] was used as the starting point. Thereafter, different model configurations were tested, where the number of layers, hidden units in

the layers, and the hyperparameters were experimented with to obtain the best results.

### 4.4.2 Loss, Optimizer and Evaluation Metrics

The loss function used during compilation was the Categorical Cross-Entropy loss function. As explained in Section 2.4.2.1, for each data point in a sample, the Categorical Cross-Entropy loss is calculated by computing formula 4.5, where $\hat{y}_i$ is the $i$-th value of the model output and $y_i$ is the actual value [27].

$$CC = L(y, \hat{y}) = \sum_{i=1}^{N} y_i \cdot \log \hat{y}_i \tag{4.5}$$

Adam optimizer algorithm was selected for the optimization algorithm. The model is trained using the training set and evaluated using the test set, with accuracy, precision, recall, f-1 score and the support are calculated. With these evaluation metrics, the confusion matrix is used to better visualize the model performance.

# Chapter 5

# Results and Discussions

This chapter presents the results acquired from the experiments based on the methodology described on the previous chapter. Section 5.1 presents a brief overview of the tools and platform used for the experiments. Section 5.2 presents the results from the autoencoder workflow while Section 5.3 the results from the fault classifier using the Convolutional Neural Network. For each of the methods, a brief overview of the data processing will be presented.

## 5.1 Overview

All of the work was done using a MacBook Pro Late 2011 with a 2.8 GHz Inter Core i7 processor. The model training was done using Google Colab for use of Jupyter Notebooks and free GPU access. Python, Tensorflow, Keras, Scikit-Learn, among other libraries listed in Appendix A, were used throughout the thesis for data preprocessing and visualization. The public Python package cwru_py3 [100] was used for splitting and shuffling the data in the CWRU dataset. A modified version of such package, hereinafter referred to as knightec_py3, was adapted for the Ametller dataset.

As described in Section 2.4.2, the batch size is the number of samples used during one full pass during the network training (forward/backward propagation of derivatives), and the learning rate sets how much to change the model's weights. The batch size can range from 1 sample (Stochastic Gradient Descent) all the way to the total number of samples in the dataset (Batch Gradient Descent). Any value in between would be referred to as Mini-Batch Gradient Descent. The learning rate can range between 0.00

and 1.00.

Keras Tuner [99], which works by running the training and evaluation datasets with different hyperparameters and then selecting the best performing parameters, was used to find the ideal batch size and learning rate. Ultimately, Keras' default parameters of a batch size of 32 samples per training cycle and a $0.001$ learning rate were used.

The size of the samples, $M$, was selected in reference to [4]. Two different values of $M$ were tested, $M = 16$ and $M = 64$. The proposed predictive maintenance methods were designed with size and performance in mind, such that it could potentially be run in an embedded device after conversions to microcontroller compatible format.

## 5.2 Anomaly Detection with Autoencoder

### 5.2.1 Data Processing

As detailed in Section 4.2.1, the CWRU dataset was preprocessed with Min-Max Normalization with a custom made function for scaling the time-series values between [0,1], which represents the intensity of the image pixel.

### 5.2.2 Results

As described in Section 4.3, the autoencoder model is trained using only normal data samples, with the purpose of learning to replicate the latent features of the normal behavior. Many configurations of different number of layers and hidden units were explored, with Table 5.2.3 listing them. The autoencoder method was tested with the CWRU dataset using the drive-end vibration data. Given time-constraints and proof-of-concept objective, only data at 1797 RPM was utilized. However, such method could be extended to include different speed profiles, as well as use with the Ametller dataset and tested with different types of sensors. After training the model, the loss achieved was $0.0026$ on training set, and $0.0204$ on validation set (Figure 5.2.1).

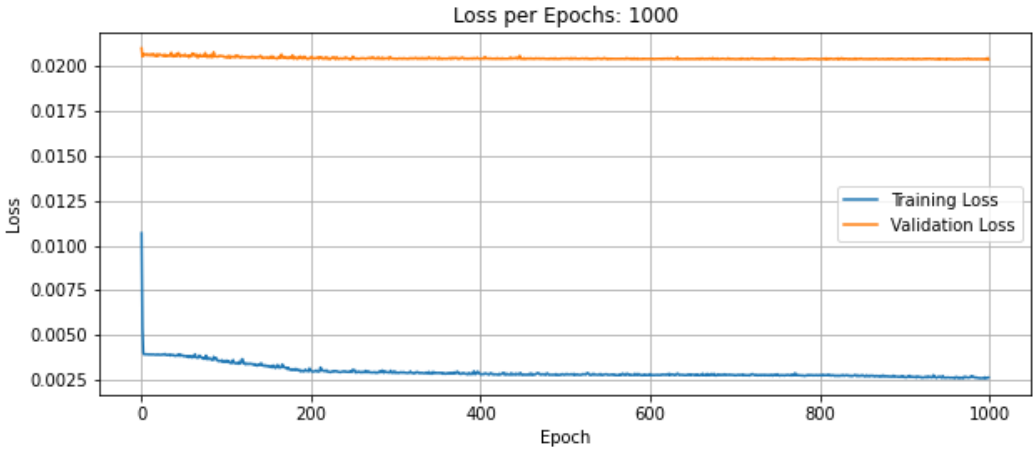
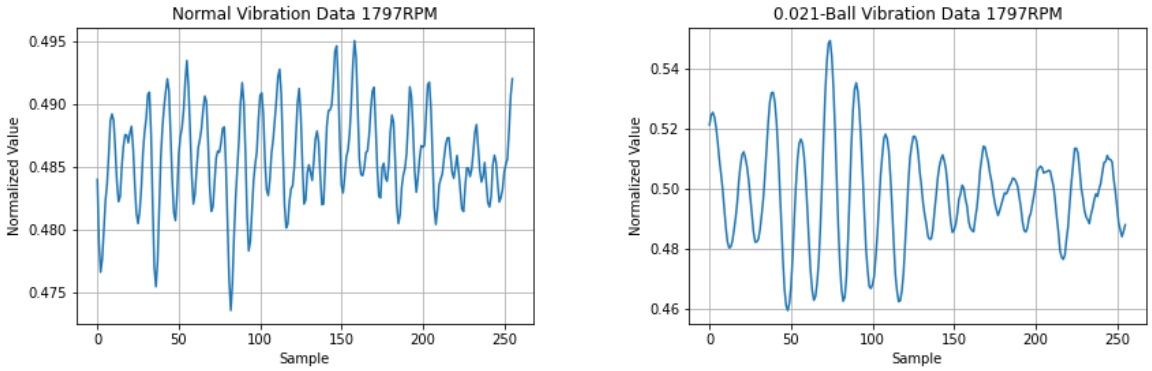



Figure 5.2.1: Autoencoder Loss

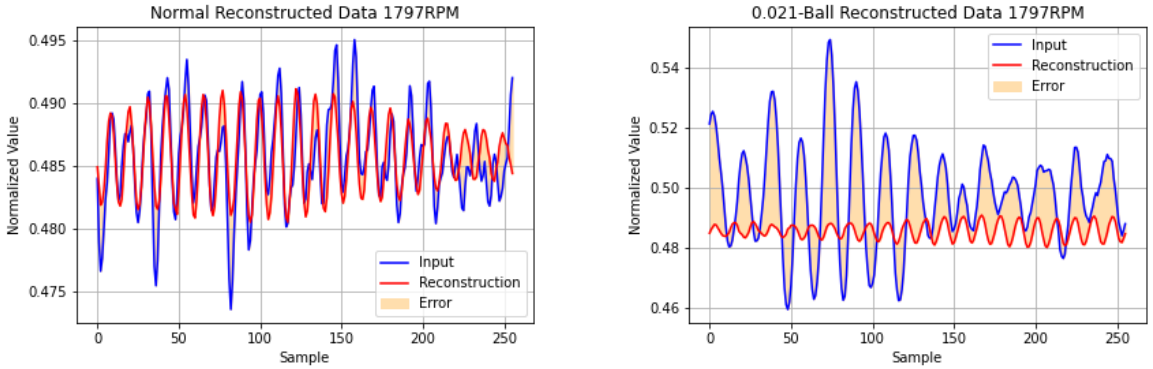
The data samples are then compared with the model's predicted reconstructed output. To demonstrate the proposed method, a normal and an anomalous samples with a size of $256$ data points, where $M = 16$ such that $M$x$M = 256$ are presented. Figure 5.2.2a and Figure 5.2.2b allow for visualization of such data samples.




(a) Normal Sample (b) Anomalous Sample

Figure 5.2.2: Vibration Data

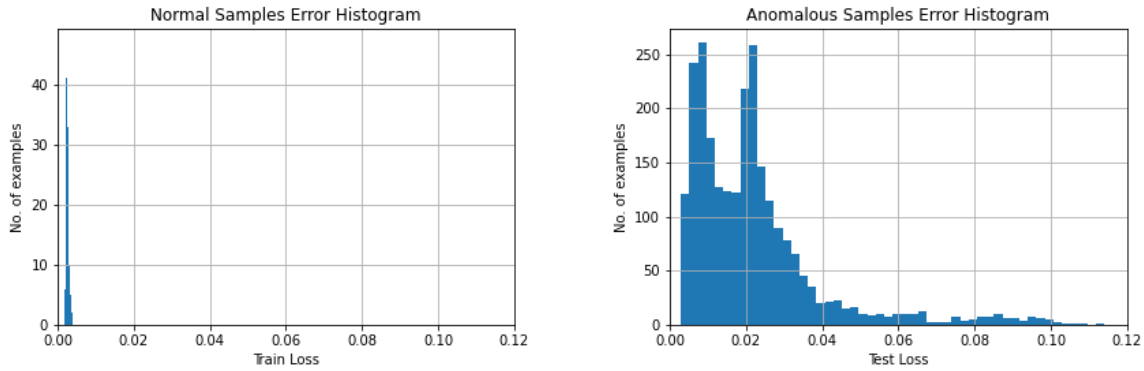
The original test sample is compared with the reconstructed decoded data (learnt latent representation) from the autoencoder. Figure 5.2.3a uses normal vibration data as input to the autoencoder. The reconstruction error (orange) is the gap between the decoded data (red) and the actual input data (blue). Figure 5.2.3b is using faulty vibration data, and as seen, the reconstructed data from the autoencoder when using anomalous data is not very good, and thus it has high reconstruction error.




(a) Reconstructed Normal Sample (b) Reconstructed Anomalous Sample

Figure 5.2.3: Reconstructed Vibration Data

This process of calculating the reconstruction error is performed to all of the samples in the test set. Figure 5.2.4 shows the histograms of the calculated reconstruction error for all the normal samples (a) and for all the anomalous data samples (b). It is evident from Figure 5.2.4, given that both histograms have the same x-axis range, that the reconstruction error for the anomalous samples is higher than that of the normal samples.




(a) Normal Data (b) Anomalous Data

Figure 5.2.4: Reconstruction Error Loss Histograms

Figure 5.2.5 combines the histograms shown in Figure 5.2.4, where blue represents the normal samples and orange the anomalous samples' loss. Anomalies can then be detected by selecting an error threshold that can distinguish between normal and anomalous sample, in other words, by calculating whether the reconstruction loss is greater than a fixed threshold (Figure 5.2.5).

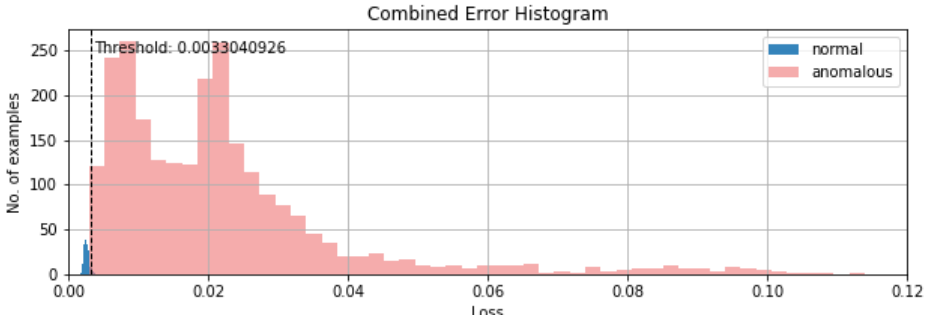



Figure 5.2.5: Combined Loss Histograms

The threshold should be selected depending on the application at hand, considering the tolerance for false positives and false negatives. A false positive corresponds to incorrectly predicting a normal condition when it is actually an anomalous sample, while a false negative corresponds to incorrectly predicting an anomalous condition when actual value is a normal condition. **In medical applications, for example, incorrectly predicting a disease can have catastrophic implications.** For this thesis, a threshold of two standard deviations from the mean of the normal data loss was used to distinguish between normal and anomalous conditions. Such a threshold was selected on a trial and error basis based on the model's prediction accuracy. Scikit-Learn's precision_recall_fscore_support package [101] was then used to evaluate the model, with the results illustrated in Table 5.2.1 with its respective confusion matrix (Figure 5.2.6) generated using Seaborn's Heatmap library function [102].

Table 5.2.1: Autoencoder Evaluation Metrics

| | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|
| **False** | 0.999 | 0.997 | 0.998 | 2489 |
| **True** | 0.966 | 0.991 | 0.979 | 232 |
| | | | | |
| **accuracy** | | | 0.996 | 2721 |
| **macro avg** | 0.983 | 0.994 | 0.988 | 2721 |
| **weighted avg** | 0.996 | 0.996 | 0.996 | 2721 |

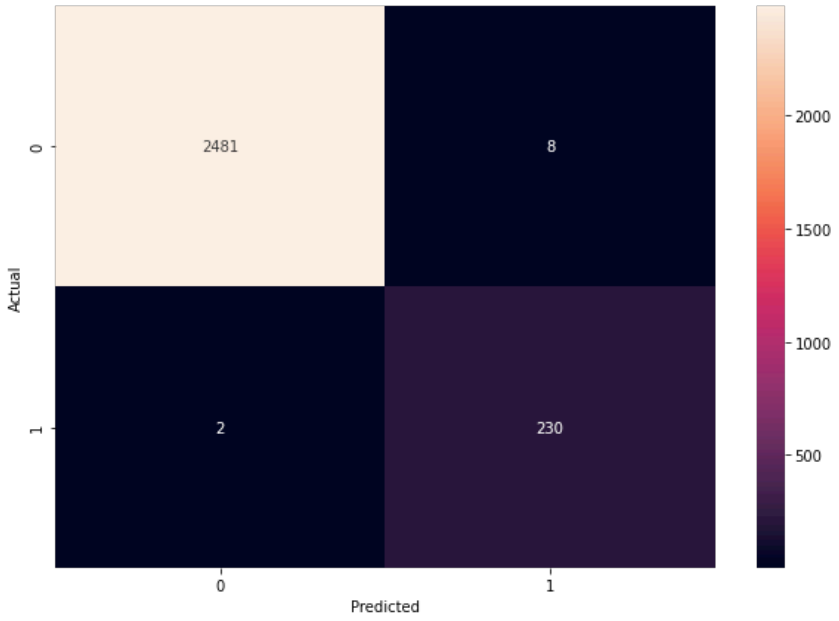



Figure 5.2.6: Confusion Matrix

These results, however, are dependent on the selected threshold. The ROC curve (Figure 5.2.7) allows us to visualize such trade-offs in the threshold selection. Increasing the threshold can cause the model to predict false positives and decreasing the threshold can cause the model to predict false negatives, directly impacting the overall accuracy, precision and recall of the system. For this reason, the AUC, a more appropriate indicator of model performance as explained in Section 2.4.3, was calculated for every model configuration.

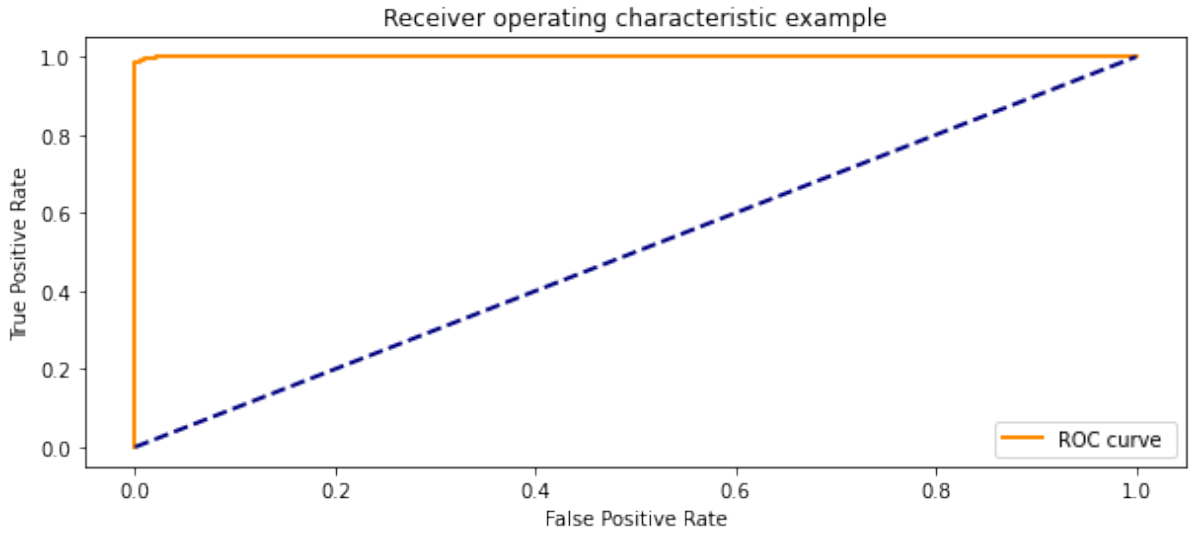



Figure 5.2.7: Receiver Operating Characteristics Curve (ROC)

As specified in Section 4.3, experiments were run in order to compare results on the different number of layers in the encoder/decoder modules and the hidden units in each layer, as well as hyperparameter tuning such as the learning rate and batch size. A size of $M = 16$ such that the data samples are of size $M$x$M = 256$ was used.

The overall strategy consisted of starting with the smallest possible model with the default Keras hyperparameters (learning rate = 0.001, batch size = 32), training for 1000 epochs, while gradually incrementing the number of hidden units and layers to the model and comparing the metrics and the model size. An exhaustive search was done to find the best model in terms of compromise between size and performance. Table 5.2.3 lists the experimented models with different layer sizes and varying hidden units. For example, the entry "64, 8, 256" would mean a three layered-model with 64 units in the first layer, 8 in the middle layer and 256 in the last layer.

Table 5.2.3: Autoencoder Model Experiments

| **Layers** | **Learning Rate** | **Batch Size** | **Accuracy** | **Parameters** |
|---|---|---|---|---|
| 8, 8, 256 | 0.001 | 32 | 0.97317 | 4,432 |
| 16, 8, 256 | 0.001 | 32 | 0.97317 | 6,552 |
| 32, 8, 256 | 0.001 | 32 | 0.98897 | 10,792 |
| 64, 8, 256 | 0.001 | 32 | 0.99632 | 19,272 |
| 128, 8, 256 | 0.001 | 32 | 0.97170 | 36,232 |
| 256, 8, 256 | 0.001 | 32 | 0.96251 | 70,152 |
| 64, 16, 8, 16, 256 | 0.001 | 32 | 0.96251 | 22,120 |
| 32, 16, 8, 16, 256 | 0.001 | 32 | 0.96251 | 13,384 |
| 256, 64, 8, 64, 256 | 0.001 | 32 | 0.96251 | 99,976 |

After selecting a tentative final model from the initial layer and hidden unit experiments, hyperparameter tuning was performed. Table 5.2.5 lists the different configurations tested along with the respective hyperparameters used, the accuracy achieved and the number of parameters. For embedded purposes and microcontroller use, the number of parameters is crucial given the limited memory and resource constraints.

Table 5.2.5: Autoencoder Hyperparameter Fine-Tuning

| **Layers** | **Learning Rate** | **Batch Size** | **Accuracy** | **Parameters** |
|---|---|---|---|---|
| 64, 8, 256 | 0.001 | 32 | 0.99632 | 19272 |
| 64, 8, 256 | 0.001 | 64 | 0.99522 | 19,272 |
| 64, 8, 256 | 0.001 | 128 | 0.99338 | 19,272 |
| 64, 8, 256 | 0.001 | 16 | 0.98934 | 19,272 |
| 64, 8, 256 | 0.001 | 256 | 0.98897 | 19,272 |
| 64, 8, 256 | 0.001 | 512 | 0.98823 | 19,272 |
| 64, 8, 256 | 0.01 | 32 | 0.96251 | 19,272 |
| 64, 8, 256 | 0.0001 | 32 | 0.98860 | 19,272 |
| 64, 8, 256 | 0.1 | 32 | 0.96324 | 19,272 |

The best performing model architecture is detailed in Table 5.2.7, where the hidden units, activation functions, and number of parameters are specified.

Table 5.2.7: Autoencoder Best Model Specifications

| **Layer** | **Hidden Units** | **Activation** | **Param #** | **Description** |
|---|---|---|---|---|
| dense | 64 | relu | 14912 | encoder |
| dense | 8 | relu | 2056 | bottleneck |
| dense | 256 | sigmoid | 2304 | decoder |
| Total params: 19272<br>Accuracy: 99.63248 | | | | |

Table 5.2.9: Evaluation Metrics

| | |
|---|---|
| **Layers** | 64, 8, 256 |
| **Accuracy** | 0.99632 |
| **Precision** | 0.99593 |
| **Recall** | 0.99595 |
| **AUC** | 0.99981 |
| **Parameters** | 19272 |

### 5.2.3 Discussion

The changes in hidden units were done in increments of $2^n$, the hidden layer was kept at 8 hidden units, and the output layer must always match the size of the data we are reconstructing, which in this case is 256. As illustrated by Table 5.2.3, all of the tested models achieve an accuracy higher than $96\%$. For this specific application, a model with deeper architecture does not translate into better performance. A deeper architecture might be beneficial when dealing with more complex, non-linear functions [14], however, in this case the model with the highest accuracy contained only 1 hidden layer and was enough to achieve best results.

The hyperparameter optimization was performed using Keras Tuner's [99] Hyperband [103] approach, which uses "Bayesian optimization to adaptively select configurations", along with non-stochastic resource allocation and early-stopping. Table 5.2.5 illustrates the different configurations used and the results achieved where it is evident that different values for the learning rate have more of an impact to the overall model performance than that of batch size update.

The trade-off between the model size and performance must be chosen depending on the application, development board and platform at hand, dependent on the memory, power, and processing constraints. For purposes of the thesis, and given that the training was performed using Google Collab, the selected model was that with the best accuracy, where such model underwent hyperparameter optimization for fine-tuning. Table 5.2.5 lists the workflow for the hyperparameter experiments.

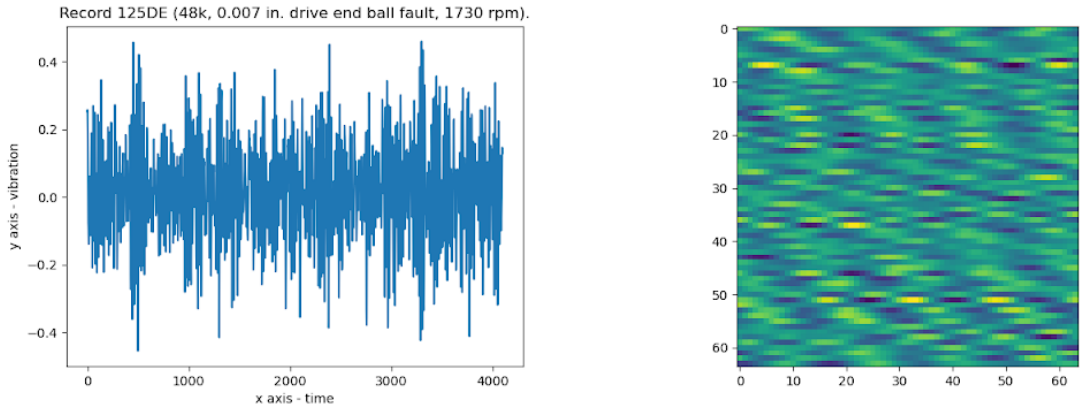
## 5.3 Fault Classifier with CNN

### 5.3.1 Data Processing

As previously described in Section 4.4.1, a CNN model based on LeNet-5 [4] was used as the starting point, composed of two convolution layers, two pooling layers, and three fully connected layers, and utilizing the tanh activation function [104]. Thereafter, different model configurations were tested, where the number of layers, hidden units in the layers, and the hyperparameters were experimented with to obtain the best results. Both datasets were preprocessed with Min-Max Normalization, scaling the time-series values between [0,1], which represents the intensity of the image pixel.

The Python package cwru_py3 [100] was used for splitting and shuffling the data in the CWRU dataset. A modified version of such package, hereafter referred to as knightec_py3, was adapted for the Ametller dataset.

For the CWRU Dataset case, Figure 5.3.1 shows a sample in time series form (Fig. 5.3.1a) and its respective converted $M \times M$ image (Fig. 5.3.1b). The intensity of the color represent the normalized value where a lighter pixel corresponds to a higher vibration value.




(a) Data (b) Image

Figure 5.3.1: Vibration Data to 2-D Image

The conversion process is applied to all of the samples in the dataset and then plotted and displayed using Matplotlib's PyPlot and ImageGrid toolkits [105]. Figure 5.3.2 shows the constructed 2-D image for each of the different fault classes (Figure 5.3.3).

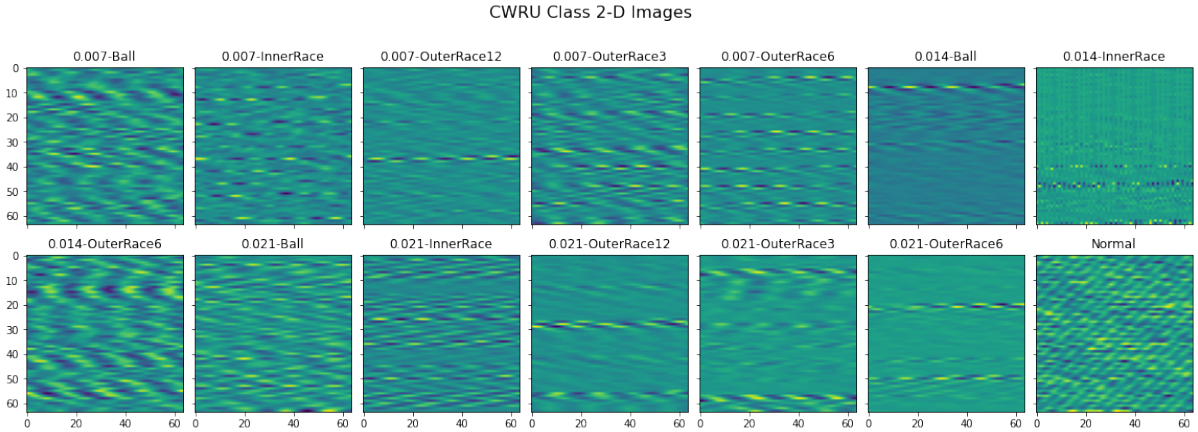



Figure 5.3.2: 2-D Images For All Faults

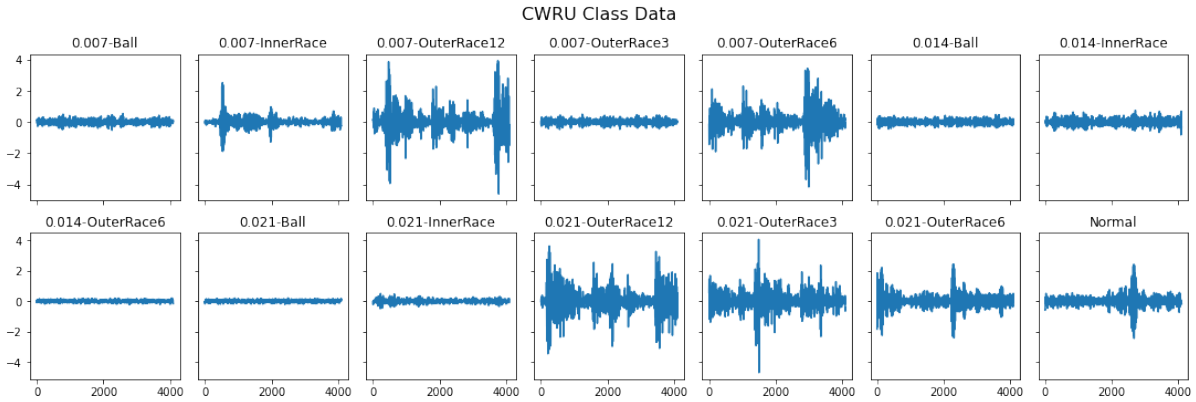



Figure 5.3.3: Respective Time Series Data For All Faults

Similarly, in the case of the Ametller dataset, Figure 5.3.4 shows the 2-D images recreated from the respective time series (Figure 5.3.5) for each of the different fault classes.

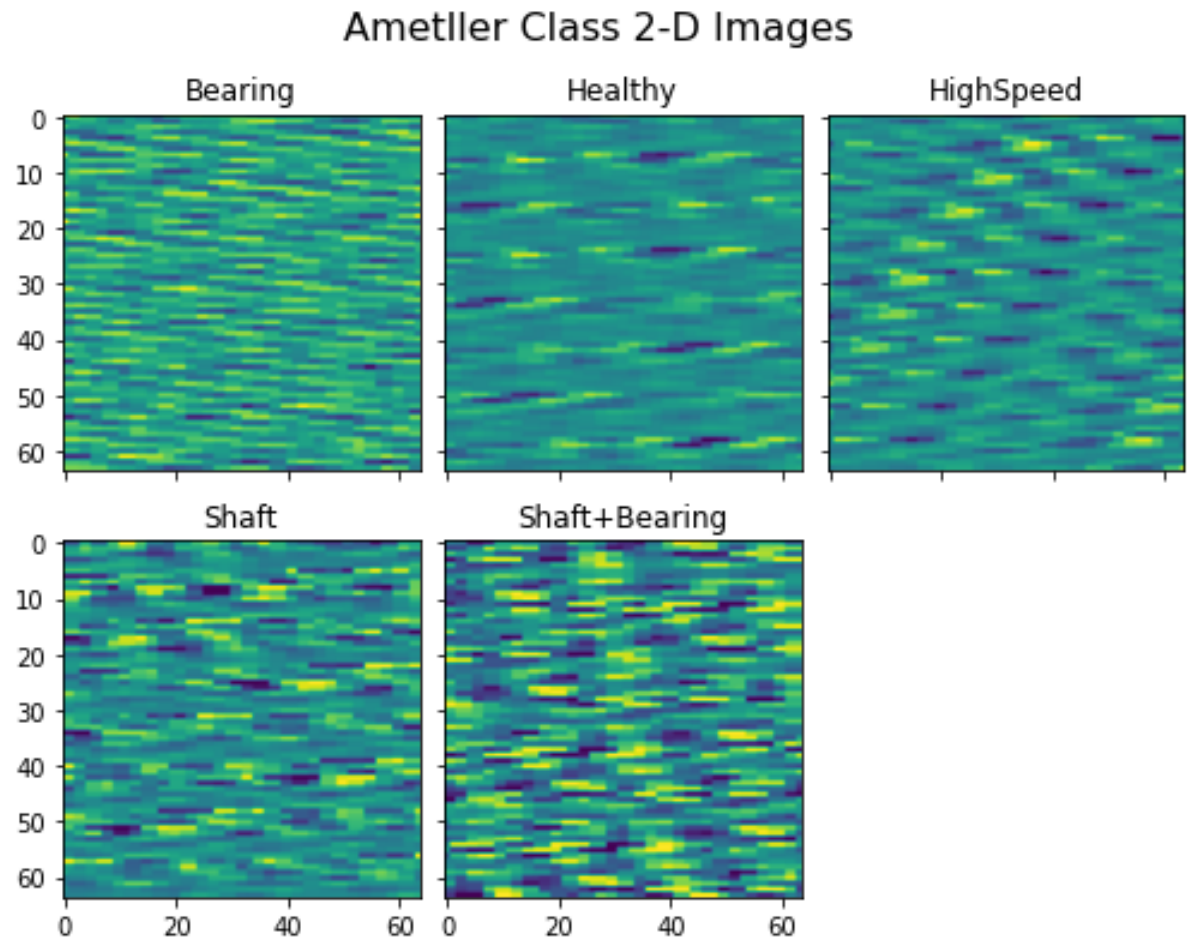



Figure 5.3.4: 2-D Images For All Faults

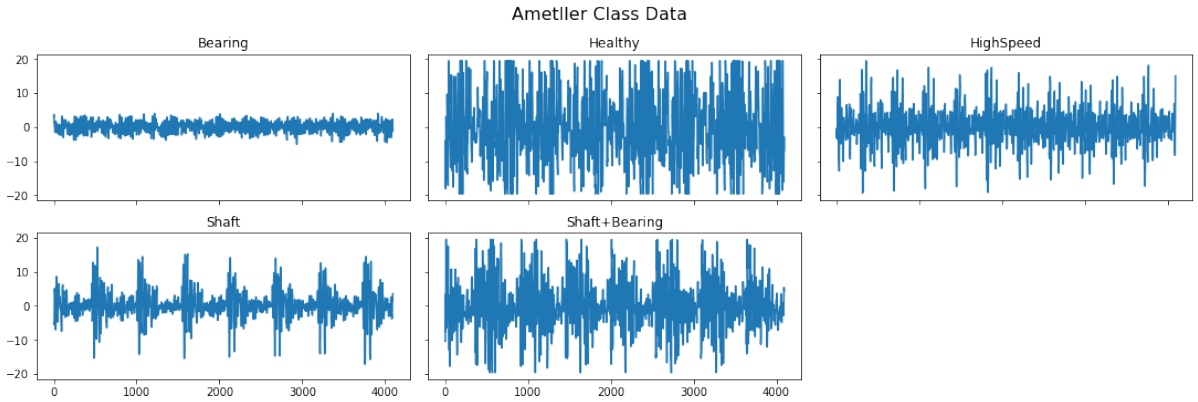
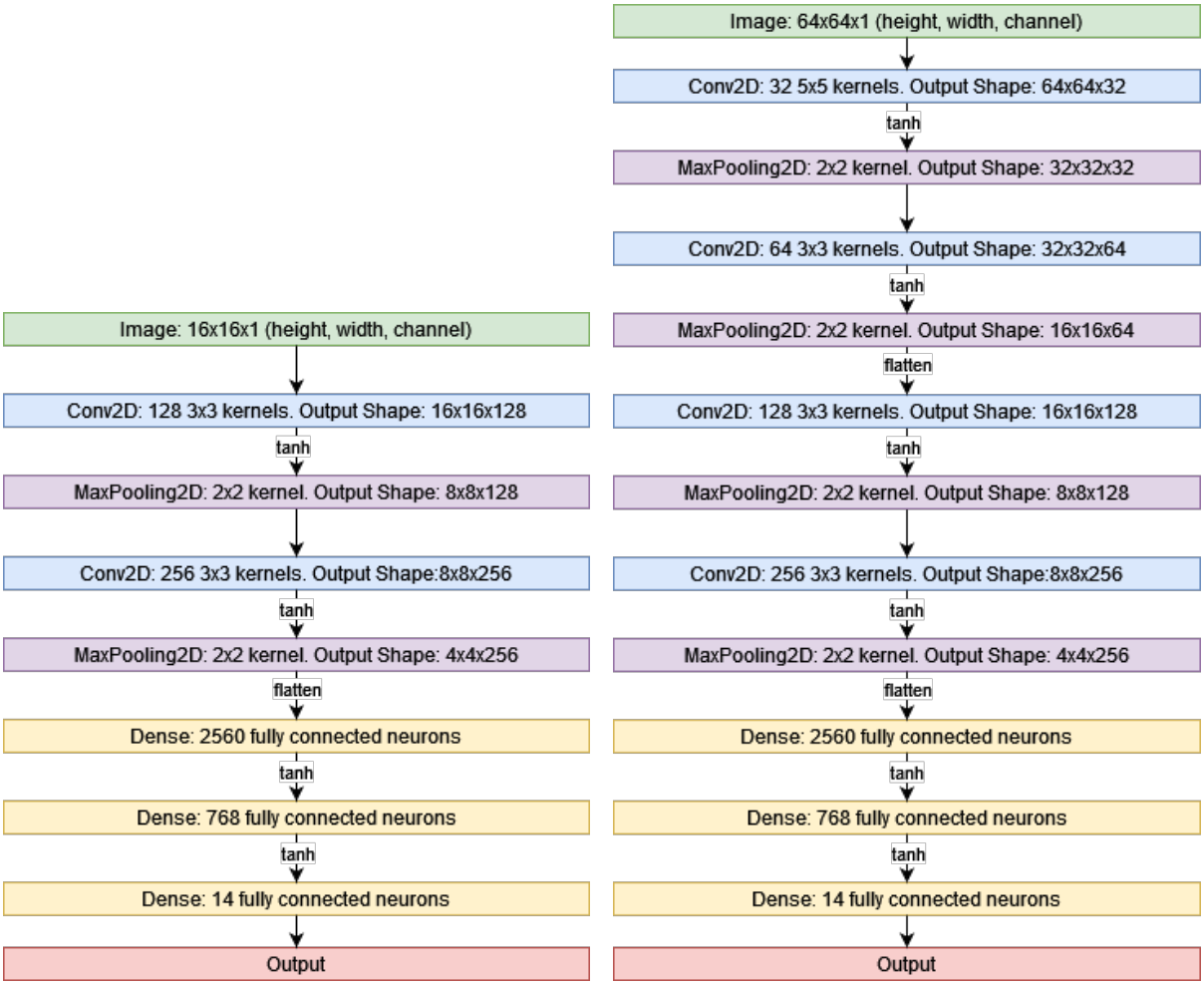



Figure 5.3.5: Respective Time Series Data For All Faults

In addition to the time series to 2-D image conversion, initial versions of the Ametller dataset were class imbalanced, with the number of samples being 458, 404, 45, 45, and 402 for each of the five classes. This imbalanced dataset problem was solved by using the up-weighting method provided by Scikit-Learn's class_weight package [101]. Later versions of the dataset had the same number of samples per class, which resulted in a balanced dataset.

### 5.3.2 Results

The workflow was first created and tested on the CWRU dataset, and later adapted to the Ametller dataset given that the final version of the dataset was not available until late in the term. The CWRU drive-end accelerometer vibration data at 1797 RPM was tested with $M = 16$ and $M = 64$, which affects the sample size and thus, the model architecture, the number of trainable parameters, and the number of image samples (dataset size). Figure 5.3.6 presents the LeNet-based model architecture used for the experiments.




(a) M=16 (b) M=64

Figure 5.3.6: CNN LeNet-Based Models

Table 5.3.1 shows the accuracy and parameter results from using the LeNet-based model architecture described previously, with the highest accuracy obtained when $M = 16$. The difference in the number of trainable parameters between both models is due to the first four layers that are present in the $M = 64$ (layers used to adapt to the input image size).

Table 5.3.1: CWRU Accuracy Metric

| Dataset | M | Accuracy | Parameters |
|---|---|---|---|
| 48kHz | 64 | 0.865 | 12,855,824 |
| 48kHz | 16 | 0.786 | 12,763,920 |
| 12kHz | 64 | N/A | 12,855,824 |
| 12kHz | 16 | 0.945 | 12,763,920 |

The methodology with the same model architectures were then tested on the final version of the Ametller dataset. Additional to the 3-axis accelerometer vibration data, the Ametller dataset also contains 3 current sensors, 1 piezoelectric vibration data, and 2 temperature sensors. Given the poor results obtained with the temperature sensors, the temperature data was omitted from the following results. For each of the sensor data, the Ametller dataset was tested with the same workflow and model configurations. Table 5.3.3 lists the accuracy results from running the LeNet-based model experiments. The number after the 'Current' sensor refers to the electrical phase measured, sensor 'Vibration 1' refers to the piezoelectric sensor, and 'Vibration 2' refers to the 3-axis accelerometer sensor, where the 'x, y, and z' labels refer to the axis.

Table 5.3.3: Single Sensor Experiment's Accuracy

| **Sensor** | **M=16** | **M=64** |
|---|---|---|
| Current 1 | 0.660 | 0.569 |
| Current 2 | 0.650 | 0.585 |
| Current 3 | 0.673 | 0.636 |
| Vibration 1 | 0.691 | 0.613 |
| Vibration 2x | 0.961 | 0.990 |
| Vibration 2y | 0.965 | 0.994 |
| Vibration 2z | 0.947 | 0.994 |

Given that the vibration sensor data achieved the best results in the preliminary experiments, further fine-tuning and configurations were performed, with the results shown in Table 5.3.5. For all experiments, a learning rate of 0.0002 and 1000 epochs were used. The batch size was 512 for $M = 16$ experiments and 32 for $M = 64$ experiments. For experiments 1 through 3, vibration axis data was used separately at a single rpm of 100. Experiment 4 combines all the different rpm data into 1-channel images. Experiment 5 concatenates the x, y, and z axis accelerometer vibration data into 1-channel at 100 rpm. Lastly, experiment 6 fuses the x, y, and z axis accelerometer vibration data into 3-channel images, where each sensor axis corresponds to a channel.

Table 5.3.5: Vibration Experiment's Accuracy

| Experiment | Sensor | RPM | Channels | M=16 | M=64 |
|---|---|---|---|---|---|
| 1 | X - only | 100 | single | 0.961 | 0.990 |
| 2 | Y - only | 100 | single | 0.965 | 0.994 |
| 3 | Z - only | 100 | single | 0.947 | 0.994 |
| 4 | X - only | all | single | 0.930 | 0.991 |
| 5 | X, Y, Z - concatenate | 100 | single | 0.935 | 0.990 |
| 6 | X, Y, Z - fusion | 100 | three | 0.995 | 0.980 |

### 5.3.3 Discussion

The fault classifier with a convolutional neural network is able to detect the root cause of the problem with great performance. As illustrated in Table 5.3.5, the results for all experiments are above $93\%$ accuracy for all experiments, however, it is evident that the results for $M = 64$ were overall higher than when $M = 16$ was used.

The selection of $M$ has great effect on the size of the images and the dataset size, as well as a great impact on the the model architecture itself. With images of size $(4096x4096)$, the convolutional neural network has more information to learn from, compared to images of size $(256x256)$. Such differences in image size could be a reason for the difference in performance. Furthermore, the size of the dataset itself is also a very important contributing factor on model performance. With $M = 64$, the dataset contains $1,000$ image samples, while the dataset is of size $21,000$ when $M = 16$. The influence of the size of the dataset was evident with initial versions of the Ametller dataset which contained considerably less data samples and therefore did not perform as well. With different values of $M$, the number of layers changes to adapt to the input images as more layers are needed to process bigger images. It is interesting to note that the number of parameters did not considerably increase when using a deeper model to adapt using $M = 64$. One reason for this could be the high number of connections in the last Fully Connected layer after the pooling layer when using $M = 16$.

Regarding the sensor performance, from the all the experiments performed with different types of sensors, the vibration data from the 3-axis accelerometer gave the best results. The high frequency content of the vibration data allows for feature-rich images that are adequate for fault classification using convolutional neural networks

(the data acquisition system samples at 5.37kHz and the 3-axis accelerometer sensor (Adafruit LIS3DH) has a frequency range of up to 5kHz). Given the increased widespread use of the MCSA for fault diagnosis methods, good results were expected from the use of current data. However, opposed to high-frequency vibration data, and although the current sensor used (Allegro MicroSystems ACS723) has frequency bandwidth of 20kHz, current supply itself has a preimposed frequency of 50-60Hz with the high frequency components being noise. For this reason, current data seems to be better suited for signal-processing methods. Provided that the piezoelectric vibration sensor (TE Connectivity 1006015-1) is limited to low frequencies up to 40Hz, it is not a good option for this method. The data obtained from such sensor contains no noticeable variations at high frequencies and does not match with the type of detection we designed. The temperature sensor (Analog Devices AD8495) was used in initial tests but was subsequently set aside given its poor performance. For this kind of methodology and data acquisition setup, temperature data does not seem to be a good choice. Temperature changes are gradual and of low frequency, characteristics not appropriate for high-speed data acquisition.

# Chapter 6

# Conclusions

With the rapid development and adoption of IoT in smart factories, the amount of data collected is swiftly increasing and with it the demand for effective data-driven methods. The aim of this thesis was to explore predictive maintenance and fault diagnosis methodologies using different deep learning data-driven techniques that can incorporate the feature selection and diagnosis in a single step, eliminating the need of field-specific expert knowledge.

Deep learning methods can leverage the use of historical data and data acquired by sensors in smart industries and allow for prediction models using raw data without the need of industry-specific knowledge and feature engineering. The anomaly detection with an autoencoder method proposes a good solution for manufacturers that do not have historical failure data of the equipment. By retrofitting existing equipment [80], it is possible to collect data of equipment running in normal operation until failure. Any deviation from the baseline behavior could be flagged for further inspection and labeled as an anomaly for future detection. As seen in the results section, the proposed anomaly detection offers good performance using a small neural network of only three layers. Additionally, the fault classifier with a convolutional neural network was able to detect the root cause of the problem with great performance. Such classification mechanism can be implemented when a factory has collected enough historical data for fault diagnosis in a supervised manner.

With the development of machine learning platforms, such as Tensorflow Lite [106], and the development of AI accelerators that would allow running machine learning natively in microcontrollers, these deep learning methods will offer a good solution to

fault detection and predictive maintenance.

As seen from the results in Section 5.3, the vibration sensor data is the most effective type of data given their high-frequency components. The current and piezoelectric data's lower frequency information (60Hz for current and 40Hz for piezo) did not perform as well in this type of fault diagnosis methodology. Temperature data was not utilized given their poor results in initial experiments.

### 6.0.1 Limitations

The biggest limitation throughout the course of the degree project was the lack of technical guidance and mentorship in terms of data science and project methodology. When running into a question or a doubt, the only resource available was books and online tutorials. Additionally, the lack of a dataset and initial project goal caused the project to change course multiple times throughout the semester.

### 6.0.2 Future Work

Because of time constraints or objective scope, there are several items whose implementation and exploration is of interest. The autoencoder method could also be extended to the Ametller dataset and tested with different sensor types such as current and temperature, as well as performing a pre-processing step with signal-processing methods such as FFT to combine signal-based and data-driven fault diagnosis methods. As listed in Table 5.3.1, the number of parameters for both models ($M = 16$ and $M = 64$) resulted in a similar number of parameters. The use of quantization, pruning, and weight clustering can be implemented with the aim of compressing and reducing the size of the model [106]. Lastly, the use of a web-based application such as [107] would be a very convenient for factory technicians and equipment owners.

#### 6.0.2.1 Remaining Useful Life

The estimation of the Remaining Useful Life (RUL) is great interest and use for the PHM of equipment. The RUL, as the name implies, is the time left an equipment has until a fault or deviation from expected normal operating conditions [108] occurs. RUL would allow for the most efficient maintenance procedures. Accurate estimation of the RUL, however, is a complex process that is in process of discovery by industry and

researchers, and a place where the use of Deep Learning can be used with great results. For example, different Deep Learning networks have been used for RUL estimation such as CNNs [1, 109, 110], LSTMs [111–113], and Deep Belief Networks [114] using NASA's C-MAPPS data set [115], a commonly used benchmark RUL dataset.

# Bibliography


[1] Jiang, J. and Kuo, C. “Enhancing Convolutional Neural Network Deep Learning for Remaining Useful Life Estimation in Smart Factory Applications”. In: *2017 International Conference on Information, Communication and Engineering (ICICE)*. Nov. 2017, pp. 120–123. DOI: 10.1109/ICICE.2017.8478928.

[2] Çınar, Zeki Murat, Abdussalam Nuhu, Abubakar, Zeeshan, Qasim, Korhan, Orhan, Asmael, Mohammed, and Safaei, Babak. “Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0”. en. In: *Sustainability* 12.19 (Jan. 2020). Number: 19 Publisher: Multidisciplinary Digital Publishing Institute, p. 8211. DOI: 10.3390/su12198211. URL: https://www.mdpi.com/2071-1050/12/19/8211 (visited on 08/13/2021).

[3] Chen, Kun-Chih and Gao, Zi-Jie. “Integrated Group-based Valuable Sensor Selection Approach for Remaining Machinery Life Estimation in the Future Industry 4.0 Era”. In: *2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. ISSN: 2472-9124. Aug. 2020, pp. 1–4. DOI: 10.1109/VLSI-DAT49148.2020.9196260.

[4] Wen, Long, Li, Xinyu, Gao, Liang, and Zhang, Yuyan. “A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method”. In: *IEEE Transactions on Industrial Electronics* 65.7 (July 2018). Conference Name: IEEE Transactions on Industrial Electronics, pp. 5990–5998. ISSN: 1557-9948. DOI: 10.1109/TIE.2017.2774777.

[5] Miljković, Dubravko. “Brief Review of Motor Current Signature Analysis”. In: *CrSNDT Journal* 5 (June 2015), pp. 14–26.

[6] Barnes, Malcolm. *Practical Variable Speed Drives and Power Electronics*. en. Google-Books-ID: LxW9F9WCixcC. Elsevier, June 2003. ISBN: 978-0-08-047391-8.

[7] Lee, Y. O., Jo, J., and Hwang, J. "Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection". In: *2017 IEEE International Conference on Big Data (Big Data)*. Dec. 2017, pp. 3248–3253. DOI: 10.1109/BigData.2017.8258307.

[8] Paolanti, Marina, Romeo, Luca, Felicetti, Andrea, Mancini, Adriano, Frontoni, Emanuele, and Loncarski, Jelena. "Machine Learning approach for Predictive Maintenance in Industry 4.0". In: *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. July 2018, pp. 1–6. DOI: 10.1109/MESA.2018.8449150.

[9] Lee, Jong-Hyun, Pack, Jae-Hyung, and Lee, In-Soo. "Fault Diagnosis of Induction Motor Using Convolutional Neural Network". In: *Applied Sciences* 9 (July 2019), p. 2950. DOI: 10.3390/app9152950.

[10] Gao, Zhiwei, Cecati, Carlo, and Ding, Steven X. "A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part II: Fault Diagnosis With Knowledge-Based and Hybrid/Active Approaches". In: *IEEE Transactions on Industrial Electronics* 62.6 (June 2015). Conference Name: IEEE Transactions on Industrial Electronics, pp. 3768–3774. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2419013.

[11] Benbouzid, M.E.H. and Kliman, G.B. "What stator current processing-based technique to use for induction motor rotor faults diagnosis?" In: *IEEE Transactions on Energy Conversion* 18.2 (June 2003). Conference Name: IEEE Transactions on Energy Conversion, pp. 238–244. ISSN: 1558-0059. DOI: 10.1109/TEC.2003.811741.

[12] Gao, Zhiwei, Cecati, Carlo, and Ding, Steven X. "A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches". In: *IEEE Transactions on Industrial Electronics* 62.6 (June 2015). Conference Name: IEEE Transactions on Industrial Electronics, pp. 3757–3767. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2417501.

[13] Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. Cambridge, United States: MIT Press, 2012. ISBN: 978-0-262-30524-2. (Visited on 07/14/2021).

[14] Chollet, François. *Deep learning with Python*. eng. 1st edition. Shelter Island, NY: Manning Publications, 2018. ISBN: 978-1-61729-443-3.

[15] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/ (visited on 07/09/2021).

[16] Peltarion. *Available activations | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations (visited on 11/09/2021).

[17] Brownlee, Jason. *How to Choose an Activation Function for Deep Learning*. en-US. Jan. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (visited on 11/09/2021).

[18] Pathimind. *A Beginner's Guide to Neural Networks and Deep Learning*. en. URL: http://wiki.pathmind.com/neural-network (visited on 11/09/2021).

[19] Peltarion. *Linear activation function | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations/linear (visited on 11/09/2021).

[20] Peltarion. *ReLU activation function | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations/relu (visited on 11/09/2021).

[21] Peltarion. *Softmax activation function | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations/softmax (visited on 11/09/2021).

[22] Peltarion. *Sigmoid activation function | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations/sigmoid (visited on 11/09/2021).

[23] Peltarion. *Tanh activation function | Build an AI model*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/activations/tanh (visited on 11/09/2021).

[24] Peltarion. *What are the Optimization principles in Deep Learning*. en. 2021. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimization-principles-(in-deep-learning) (visited on 07/29/2021).

[25] Peltarion. *Regression loss metrics on the Peltarion Platform*. en. July 2021. URL: https://peltarion.com/knowledge-center/documentation/evaluation-view/regression-loss-metrics/mae-/-mean-absolute-error (visited on 07/13/2021).

[26] Peltarion. *Regression loss metrics on the Peltarion Platform*. en. URL: https://peltarion.com/knowledge-center/documentation/evaluation-view/regression-loss-metrics/mse-/-mean-squared-error (visited on 07/29/2021).

[27] Peltarion. *Categorical crossentropy loss function | Peltarion Platform*. en. 2021. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy (visited on 07/29/2021).

[28] Peltarion. *Binary crossentropy loss function | Peltarion Platform*. en. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy (visited on 11/09/2021).

[29] Brownlee, Jason. *Gradient Descent With RMSProp from Scratch*. en-US. May 2021. URL: https://machinelearningmastery.com/gradient-descent-with-rmsprop-from-scratch/ (visited on 11/09/2021).

[30] Brownlee, Jason. *Gradient Descent With AdaGrad From Scratch*. en-US. June 2021. URL: https://machinelearningmastery.com/gradient-descent-with-adagrad-from-scratch/ (visited on 11/09/2021).

[31] Kingma, Diederik P. and Ba, Jimmy. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 07/13/2021).

[32] Brownlee, Jason. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. en-US. July 2017. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 07/13/2021).

[33] Brownlee, Jason. *Tour of Evaluation Metrics for Imbalanced Classification*. en-US. Jan. 2020. URL: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/ (visited on 11/09/2021).

[34] LeCun, Yann. *MNIST Demos on Yann LeCun's website*. URL: http://yann.lecun.com/exdb/lenet/ (visited on 07/18/2021).

[35] Salih, Atheer, Gaeid, Khalaf, and Hew, Wooi. "Fault Diagnosis of Induction Motor Using MCSA and FFT". In: *Electrical and Electronic Engineering 1.2 (2011)* (Jan. 2012).

[36] Malik, Hasmat and Mishra, Sukumar. "Artificial neural network and empirical mode decomposition based imbalance fault diagnosis of wind turbine using TurbSim, FAST and Simulink". en. In: *IET Renewable Power Generation* 11.6 (2017). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-rpg.2015.0382, pp. 889–902. ISSN: 1752-1424. DOI: 10.1049/iet-rpg.2015.0382. URL: http://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rpg.2015.0382 (visited on 08/11/2021).

[37] Luo, Yefeng, Qiu, Jianqi, and Shi, Cenwei. "Fault Detection of Permanent Magnet Synchronous Motor Based on Deep Learning Method". In: *2018 21st International Conference on Electrical Machines and Systems (ICEMS)*. Oct. 2018, pp. 699–703. DOI: 10.23919/ICEMS.2018.8549129.

[38] Li, B., Chow, M.-Y., Tipsuwan, Y., and Hung, J.C. "Neural-network-based motor rolling bearing fault diagnosis". In: *IEEE Transactions on Industrial Electronics* 47.5 (Oct. 2000). Conference Name: IEEE Transactions on Industrial Electronics, pp. 1060–1069. ISSN: 1557-9948. DOI: 10.1109/41.873214.

[39] Krishna, Merugu Siva Rama and Ravi, Kiran S. "Fault diagnosis of induction motor using Motor Current Signature Analysis". In: *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)*. Mar. 2013, pp. 180–186. DOI: 10.1109/ICCPCT.2013.6528849.

[40] Badihi, Hamed, Zhang, Youmin, and Hong, Henry. "Wind Turbine Fault Diagnosis and Fault-Tolerant Torque Load Control Against Actuator Faults". In: *IEEE Transactions on Control Systems Technology* 23.4 (July 2015).

Conference Name: IEEE Transactions on Control Systems Technology, pp. 1351–1372. ISSN: 1558-0865. DOI: 10.1109/TCST.2014.2364956.

[41] Sanchez, Hector, Escobet, Teresa, Puig, Vicenç, and Odgaard, Peter Fogh. "Fault Diagnosis of an Advanced Wind Turbine Benchmark Using Interval-Based ARRs and Observers". In: *IEEE Transactions on Industrial Electronics* 62.6 (June 2015). Conference Name: IEEE Transactions on Industrial Electronics, pp. 3783–3793. ISSN: 1557-9948. DOI: 10.1109/TIE.2015.2399401.

[42] Rai, V. K. and Mohanty, A. R. "Bearing fault diagnosis using FFT of intrinsic mode functions in Hilbert–Huang transform". en. In: *Mechanical Systems and Signal Processing* 21.6 (Aug. 2007), pp. 2607–2615. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2006.12.004. URL: https://www.sciencedirect.com/science/article/pii/S0888327006002846 (visited on 08/11/2021).

[43] Wang, Y. S., Ma, Q. H., Zhu, Q., Liu, X. T., and Zhao, L. H. "An intelligent approach for engine fault diagnosis based on Hilbert–Huang transform and support vector machine". en. In: *Applied Acoustics* 75 (Jan. 2014), pp. 1–9. ISSN: 0003-682X. DOI: 10.1016/j.apacoust.2013.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0003682X13001515 (visited on 08/11/2021).

[44] Pandya, D. H., Upadhyay, S. H., and Harsha, S. P. "Fault diagnosis of rolling element bearing with intrinsic mode function of acoustic emission data using APF-KNN". en. In: *Expert Systems with Applications* 40.10 (Aug. 2013), pp. 4137–4145. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.01.033. URL: https://www.sciencedirect.com/science/article/pii/S0957417413000468 (visited on 08/11/2021).

[45] Peng, Z., Chu, F., and He, Y. "VIBRATION SIGNAL ANALYSIS AND FEATURE EXTRACTION BASED ON REASSIGNED WAVELET SCALOGRAM". en. In: *Journal of Sound and Vibration* 253.5 (June 2002), pp. 1087–1100. ISSN: 0022-460X. DOI: 10.1006/jsvi.2001.4085. URL: https://www.sciencedirect.com/science/article/pii/S0022460X01940854 (visited on 08/11/2021).

[46] Cheng, Cheng, Zhou, Beitong, Ma, Guijun, Wu, Dongrui, and Yuan, Ye. "Wasserstein Distance based Deep Adversarial Transfer Learning for

Intelligent Fault Diagnosis". In: *arXiv:1903.06753 [cs, eess, stat]* (Mar. 2019). arXiv: 1903.06753. URL: http://arxiv.org/abs/1903.06753 (visited on 08/11/2021).

[47] Wang, Xia, Liu, Changwen, Bi, Fengrong, Bi, Xiaoyang, and Shao, Kang. "Fault diagnosis of diesel engine based on adaptive wavelet packets and EEMD-fractal dimension". en. In: *Mechanical Systems and Signal Processing* 41.1 (Dec. 2013), pp. 581–597. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2013.07.009. URL: https://www.sciencedirect.com/science/article/pii/S0888327013003592 (visited on 08/11/2021).

[48] Do, Van Tuan and Chong, Ui-Pil. "Signal Model-Based Fault Detection and Diagnosis for Induction Motors Using Features of Vibration Signal in Two-Dimension Domain". en. In: *Strojniški vestnik – Journal of Mechanical Engineering* 57.09 (Sept. 2011), pp. 655–666. ISSN: 00392480. DOI: 10.5545/sv-jme.2010.162. URL: http://www.sv-jme.eu/article/signal-model-based-fault-detection-and-diagnosis-for-induction-motors-using-features-of-vibration-signal-in-two-dimension-domain/ (visited on 08/06/2021).

[49] Smith, Wade A. and Randall, Robert B. "Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study". en. In: *Mechanical Systems and Signal Processing* 64-65 (Dec. 2015), pp. 100–131. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2015.04.021. URL: https://www.sciencedirect.com/science/article/pii/S0888327015002034 (visited on 02/18/2021).

[50] J. I. Terra, M. Castelli, J. P. Fossati, M. Andrade, and Analía Conde. *Faults Detection and Remote Monitoring System for Induction Motors using MCSA Technique - PDF Free Download*. URL: https://docplayer.net/1914521-Faults-detection-and-remote-monitoring-system-for-induction-motors-using-mcsa-technique.html (visited on 08/11/2021).

[51] Legowski, S.F., Sadrul Ula, A.H.M., and Trzynadlowski, A.M. "Instantaneous power as a medium for the signature analysis of induction motors". In: *IEEE Transactions on Industry Applications* 32.4 (July 1996). Conference Name: IEEE Transactions on Industry Applications, pp. 904–909. ISSN: 1939-9367. DOI: 10.1109/28.511648.

[52] Trzynadlowski, A.M., Ghassemzadeh, M., and Legowski, S.F. “Diagnostics of mechanical abnormalities in induction motors using instantaneous electric power”. In: *IEEE Transactions on Energy Conversion* 14.4 (Dec. 1999). Conference Name: IEEE Transactions on Energy Conversion, pp. 1417–1423. ISSN: 1558-0059. DOI: 10.1109/60.815083.

[53] Pillay, P. and Xu, Z. “Motor current signature analysis”. In: *IAS ’96. Conference Record of the 1996 IEEE Industry Applications Conference Thirty-First IAS Annual Meeting* (1996). DOI: 10.1109/IAS.1996.557096.

[54] Bonaldi, E.L., Oliveira, L.Ede.L. de, Silva, L.E.B. da, and Torres, G.L. “Removing the fundamental component in MCSA using the synchronous reference frame approach”. In: *2003 IEEE International Symposium on Industrial Electronics ( Cat. No.03TH8692)*. Vol. 2. June 2003, 913–918 vol. 2. DOI: 10.1109/ISIE.2003.1267943.

[55] Douglas, H, Pillay, P, and Ziarani, A K. “A New Algorithm for Transient Motor Current Signature Analysis Using Wavelets”. en. In: (2003), p. 6.

[56] Sm, Shashidhara and Raju, Dr.P.S. “Stator winding fault diagnosis of three-phase induction motor by Park’s Vector Approach”. In: *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 2 (July 2013), pp. 2901–2906.

[57] Yin, Shen, Ding, Steven X., Xie, Xiaochen, and Luo, Hao. “A Review on Basic Data-Driven Approaches for Industrial Process Monitoring”. In: *IEEE Transactions on Industrial Electronics* 61.11 (Nov. 2014). Conference Name: IEEE Transactions on Industrial Electronics, pp. 6418–6428. ISSN: 1557-9948. DOI: 10.1109/TIE.2014.2301773.

[58] Malhi, A. and Gao, R.X. “PCA-based feature selection scheme for machine defect classification”. In: *IEEE Transactions on Instrumentation and Measurement* 53.6 (Dec. 2004). Conference Name: IEEE Transactions on Instrumentation and Measurement, pp. 1517–1525. ISSN: 1557-9662. DOI: 10.1109/TIM.2004.834070.

[59] Shen, Fei, Chen, Chao, Yan, Ruqiang, and Gao, Robert X. “Bearing fault diagnosis based on SVD feature extraction and transfer learning classification”. In: *2015 Prognostics and System Health Management Conference (PHM)*. Oct. 2015, pp. 1–6. DOI: 10.1109/PHM.2015.7380088.

[60] Susto, Gian Antonio, Schirru, Andrea, Pampuri, Simone, Mcloone, Sean, and Beghi, Alessandro. “Machine Learning for Predictive Maintenance: A Multiple Classifier Approach”. In: *Industrial Informatics, IEEE Transactions on* 11 (June 2015), pp. 812–820. DOI: 10.1109/TII.2014.2349359.

[61] Tax, David, Ypma, Alexander, and Duin, Robert. “Pump Failure Detection Using Support Vector Data Descriptions”. In: vol. 1642. Aug. 1999, pp. 415–426. ISBN: 978-3-540-66332-4. DOI: 10.1007/3-540-48412-4_35.

[62] Wijayasekara, Dumidu, Linda, Ondrej, Manic, Milos, and Rieger, Craig. “FN-DFE: Fuzzy-Neural Data Fusion Engine for Enhanced Resilient State-Awareness of Hybrid Energy Systems”. In: *IEEE Transactions on Cybernetics* 44.11 (Nov. 2014). Conference Name: IEEE Transactions on Cybernetics, pp. 2065–2075. ISSN: 2168-2275. DOI: 10.1109/TCYB.2014.2323891.

[63] Shatnawi, Yousef and Al-khassaweneh, Mahmood. “Fault Diagnosis in Internal Combustion Engines Using Extension Neural Network”. In: *IEEE Transactions on Industrial Electronics* 61.3 (Mar. 2014). Conference Name: IEEE Transactions on Industrial Electronics, pp. 1434–1443. ISSN: 1557-9948. DOI: 10.1109/TIE.2013.2261033.

[64] Chen, Zhuyun and Li, Weihua. “Multisensor Feature Fusion for Bearing Fault Diagnosis Using Sparse Autoencoder and Deep Belief Network”. In: *IEEE Transactions on Instrumentation and Measurement* PP (Mar. 2017), pp. 1–10. DOI: 10.1109/TIM.2017.2669947.

[65] Rahimilarki, R., Gao, Z., Jin, N., and Zhang, A. “Time-series Deep Learning Fault Detection with the Application of Wind Turbine Benchmark”. In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. ISSN: 2378-363X. July 2019, pp. 1337–1342. DOI: 10.1109/INDIN41052.2019.8972237.

[66] Odgaard, P. F., Stoustrup, J., and Kinnaert, M. “Fault-Tolerant Control of Wind Turbines: A Benchmark Model”. In: *IEEE Transactions on Control Systems Technology* 21.4 (July 2013). Conference Name: IEEE Transactions on Control Systems Technology, pp. 1168–1182. ISSN: 1558-0865. DOI: 10.1109/TCST.2013.2259235.

[67] Ince, T., Kiranyaz, S., Eren, L., Askar, M., and Gabbouj, M. “Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks”. In: *IEEE Transactions on Industrial Electronics* 63.11 (Nov. 2016). Conference Name: IEEE Transactions on Industrial Electronics, pp. 7067–7075. ISSN: 1557-9948. DOI: 10.1109/TIE.2016.2582729.

[68] Abdeljaber, Osama, Avci, Onur, Kiranyaz, Serkan, Gabbouj, Moncef, and Inman, Daniel. “Real-Time Vibration-Based Structural Damage Detection Using One-Dimensional Convolutional Neural Networks”. In: *Journal of Sound and Vibration* 388 (Feb. 2017), pp. 154–170. DOI: 10.1016/j.jsv.2016.10.043.

[69] Guo, Xiaojie, Chen, Liang, and Shen, Changqing. “Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis”. In: *Measurement* 93 (July 2016). DOI: 10.1016/j.measurement.2016.07.054.

[70] Jia, Feng, Lei, Yaguo, Lin, Jing, Zhou, Xin, and Lu, Na. “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data”. en. In: *Mechanical Systems and Signal Processing* 72-73 (May 2016), pp. 303–315. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2015.10.025. URL: https://www.sciencedirect.com/science/article/pii/S0888327015004859 (visited on 08/12/2021).

[71] Wen, Long, Gao, Liang, and Li, Xinyu. “A New Deep Transfer Learning Based on Sparse Auto-Encoder for Fault Diagnosis”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.1 (Jan. 2019). Conference Name: IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 136–144. ISSN: 2168-2232. DOI: 10.1109/TSMC.2017.2754287.

[72] Shao, Haidong, Jiang, Hongkai, Wang, Fuan, and Zhao, Huiwei. “An enhancement deep feature fusion method for rotating machinery fault diagnosis”. en. In: *Knowledge-Based Systems* 119 (Mar. 2017), pp. 200–220. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2016.12.012. URL: https://www.sciencedirect.com/science/article/pii/S0950705116305068 (visited on 08/12/2021).

[73] Lu, Weining, Wang, Xueqian, Yang, Chunchun, and Zhang, Tao. “A novel feature extraction method using deep neural network for rolling bearing fault diagnosis”. In: May 2015, pp. 2427–2431. DOI: 10.1109/CCDC.2015.7162328.

[74] Wang, Fengtao, Dun, Bosen, Deng, Gang, Li, Hongkun, and Han, Qingkai. "A deep neural network based on kernel function and auto-encoder for bearing fault diagnosis". In: *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)* (2018). DOI: 10.1109/I2MTC.2018.8409574.

[75] Zhang, Shen, Ye, Fei, Wang, Bingnan, and Habetler, Thomas G. "Semi-Supervised Learning of Bearing Anomaly Detection via Deep Variational Autoencoders". In: *arXiv:1912.01096 [cs, eess, stat]* (Dec. 2019). arXiv: 1912.01096. URL: http://arxiv.org/abs/1912.01096 (visited on 08/12/2021).

[76] Grbovic, Mihajlo, Li, Weichang, Subrahmanya, Niranjan A, Usadi, Adam K., and Vucetic, Slobodan. "Cold Start Approach for Data-Driven Fault Detection". In: *IEEE Transactions on Industrial Informatics* 9.4 (Nov. 2013). Conference Name: IEEE Transactions on Industrial Informatics, pp. 2264–2273. ISSN: 1941-0050. DOI: 10.1109/TII.2012.2231870.

[77] Kang, Myeongsu and Kim, Jongmyon. "Reliable Fault Diagnosis of Multiple Induction Motor Defects Using a 2-D Representation of Shannon Wavelets". In: *Magnetics, IEEE Transactions on* 50 (Oct. 2014), pp. 1–13. DOI: 10.1109/TMAG.2014.2316474.

[78] Li, Bing, Zhang, Pei-lin, Liu, Dong-sheng, Mi, Shuang-shan, Ren, Guo-quan, and Tian, Hao. "Feature extraction for rolling element bearing fault diagnosis utilizing generalized S transform and two-dimensional non-negative matrix factorization". In: *Journal of Sound and Vibration* 330 (May 2011), pp. 2388–2399. DOI: 10.1016/j.jsv.2010.11.019.

[79] Lu, Chen, Wang, Yang, Ragulskis, Minvydas, and Cheng, Yujie. "Fault Diagnosis for Rotating Machinery: A Method based on Image Processing". en. In: *PLOS ONE* 11.10 (Oct. 2016). Publisher: Public Library of Science, e0164111. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0164111. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0164111 (visited on 02/18/2021).

[80] Hesser, Daniel Frank and Markert, Bernd. "Tool wear monitoring of a retrofitted CNC milling machine using artificial neural networks". en. In: *Manufacturing Letters* 19 (Jan. 2019), pp. 1–4. ISSN: 2213-8463. DOI: 10.

1016/j.mfglet.2018.11.001. URL: https://www.sciencedirect.com/science/article/pii/S2213846318301524 (visited on 08/12/2021).

[81] CWRU. *Welcome to the Case Western Reserve University Bearing Data Center Website | Bearing Data Center*. May 2021. URL: https://web.archive.org/web/20210526191015/https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website (visited on 07/06/2021).

[82] Yang, Yanli and Fu, Peiying. "Rolling-Element Bearing Fault Data Automatic Clustering Based on Wavelet and Deep Neural Network". en. In: *Shock and Vibration* 2018 (Nov. 2018). Publisher: Hindawi, e3047830. ISSN: 1070-9622. DOI: 10.1155/2018/3047830. URL: https://www.hindawi.com/journals/sv/2018/3047830/ (visited on 03/09/2021).

[83] Afrasiabi, S., Afrasiabi, M., Parang, B., and Mohammadi, M. "Real-Time Bearing Fault Diagnosis of Induction Motors with Accelerated Deep Learning Approach". In: *2019 10th International Power Electronics, Drive Systems and Technologies Conference (PEDSTC)*. Feb. 2019, pp. 155–159. DOI: 10.1109/PEDSTC.2019.8697244.

[84] Zhang, Xiaoyuan, Liang, Yitao, Zhou, Jianzhong, and zang, Yi. "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM". en. In: *Measurement* 69 (June 2015), pp. 164–179. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2015.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0263224115001633 (visited on 07/06/2021).

[85] Vilakazi, Christina Busisiwe. "Machine Condition Monitoring Using Artificial Intelligence: The Incremental Learning and Multi-agent System Approach". en. In: (), p. 125.

[86] Lei, Yaguo, Jia, Feng, Lin, Jing, Xing, Saibo, and Ding, Steven. "An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data". In: *IEEE Transactions on Industrial Electronics* 63 (May 2016), pp. 1–1. DOI: 10.1109/TIE.2016.2519325.

[87] Shenfield, Alex and Howarth, Martin. "A Novel Deep Learning Model for the Detection and Identification of Rolling Element-Bearing Faults". en. In: *Sensors* 20.18 (Jan. 2020). Number: 18 Publisher: Multidisciplinary Digital

Publishing Institute, p. 5112. DOI: 10.3390/s20185112. URL: https://www.mdpi.com/1424-8220/20/18/5112 (visited on 07/06/2021).

[88] Hoang, Duy, Tran, Xuan, Van, Mien, and Kang, Hee-Jun. "A Deep Neural Network-Based Feature Fusion for Bearing Fault Diagnosis". In: *Sensors* 21 (Jan. 2021), p. 244. DOI: 10.3390/s21010244.

[89] Neupane, D. and Seok, J. "Bearing Fault Detection and Diagnosis Using Case Western Reserve University Dataset With Deep Learning Approaches: A Review". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 93155–93178. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990528.

[90] *Elmotorer, frekvensomriktare & växlar direkt från lager | Busck*. URL: https://www.busck.se/ (visited on 07/06/2021).

[91] *ACS55 - Small and compact drive for simple applications*. en. URL: https://new.abb.com/drives/low-voltage-ac/micro/acs55 (visited on 07/06/2021).

[92] Ioffe, Sergey and Szegedy, Christian. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". eng. In: (2015). URL: https://arxiv.org/abs/1502.03167 (visited on 07/08/2021).

[93] Raschka, Sebastian. *About Feature Scaling and Normalization*. July 2014. URL: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (visited on 07/08/2021).

[94] *Imbalanced Data*. en. URL: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data (visited on 07/08/2021).

[95] Ma, Yunqian and He, Haibo. *Imbalanced Learning: Foundations, Algorithms, and Applications*. en. John Wiley & Sons, June 2013. ISBN: 978-1-118-64633-5.

[96] Barros, Thiago M., Souza Neto, Plácido A., Silva, Ivanovitch, and Guedes, Luiz Affonso. "Predictive Models for Imbalanced Data: A School Dropout Perspective". en. In: *Education Sciences* 9.4 (Dec. 2019). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 275. DOI: 10.3390/educsci9040275. URL: https://www.mdpi.com/2227-7102/9/4/275 (visited on 07/08/2021).

[97] Zhu, Xingquan and Davidson, Ian. *Knowledge Discovery and Data Mining: Challenges and Realities*. English. Publication Title: https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59904-252-7. IGI Global, Jan. 2001. ISBN: 978-1-59904-252-7. URL: https://www.igi-global.com/gateway/book/672 (visited on 07/08/2021).

[98] Reddi, Vijay Janapa, Moroney, Laurence, and Warden, Pete. *Course | Applications of TinyML | edX*. 2021. URL: https://learning.edx.org/course/course-v1:HarvardX+TinyML2+3T2020/home (visited on 08/17/2021).

[99] O'Malley, Tom, Bursztein, Elie, Long, James, Chollet, François, Jin, Haifeng, Invernizzi, Luca, et al. *Keras Tuner*. 2019. URL: https://github.com/keras-team/keras-tuner.

[100] Jung, Juho. *cwru-py3: Case Western Reserve University Bearing Data*. URL: https://github.com/94JuHo/cwru_py3 (visited on 07/07/2021).

[101] Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, Vanderplas, Jake, Passos, Alexandre, Cournapeau, David, Brucher, Matthieu, Perrot, Matthieu, and Duchesnay, Édouard. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html (visited on 07/08/2021).

[102] Waskom, Michael L. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021). Publisher: The Open Journal, p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[103] Li, Lisha, Jamieson, Kevin, DeSalvo, Giulia, Rostamizadeh, Afshin, and Talwalkar, Ameet. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization". In: *arXiv:1603.06560 [cs, stat]* (June 2018). arXiv: 1603.06560. URL: http://arxiv.org/abs/1603.06560 (visited on 07/29/2021).

[104] Simonyan, Karen and Zisserman, Andrew. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 11/29/2021).

[105] Hunter, J. D. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007). Publisher: IEEE COMPUTER SOC, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[106] David, Robert, Duke, Jared, Jain, Advait, Reddi, Vijay Janapa, Jeffries, Nat, Li, Jian, Kreeger, Nick, Nappier, Ian, Natraj, Meghna, Regev, Shlomi, Rhodes, Rocky, Wang, Tiezhen, and Warden, Pete. “TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems”. en. In: *arXiv:2010.08678 [cs]* (Oct. 2020). arXiv: 2010.08678. URL: http://arxiv.org/abs/2010.08678 (visited on 03/08/2021).

[107] *Anomagram: Interactive Visualization of Autoencoders*. en. URL: https://anomagram.fastforwardlabs.com (visited on 07/15/2021).

[108] Saxena, Abhinav, Goebel, Kai, Simon, Don, and Eklund, Neil. “Damage propagation modeling for aircraft engine run-to-failure simulation”. In: *International Conference on Prognostics and Health Management* (Oct. 2008). DOI: 10.1109/PHM.2008.4711414.

[109] Li, Xiang, Ding, Qian, and Sun, Jian-Qiao. “Remaining useful life estimation in prognostics using deep convolution neural networks”. en. In: *Reliability Engineering & System Safety* 172 (Apr. 2018), pp. 1–11. ISSN: 0951-8320. DOI: 10.1016/j.ress.2017.11.021. URL: https://www.sciencedirect.com/science/article/pii/S0951832017307779 (visited on 08/17/2021).

[110] Sateesh Babu, Giduthuri, Zhao, Peilin, and Li, Xiao-Li. “Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life”. en. In: *Database Systems for Advanced Applications*. Ed. by Shamkant B. Navathe, Weili Wu, Shashi Shekhar, Xiaoyong Du, X. Sean Wang, and Hui Xiong. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 214–228. ISBN: 978-3-319-32025-0. DOI: 10.1007/978-3-319-32025-0_14.

[111] Wu, Yuting, Yuan, Mei, Dong, Shaopeng, Lin, Li, and Liu, Yingqi. “Remaining useful life estimation of engineered systems using vanilla LSTM neural networks”. en. In: *Neurocomputing* 275 (Jan. 2018), pp. 167–179. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.05.063. URL: https://www.sciencedirect.com/science/article/pii/S0925231217309505 (visited on 08/17/2021).

[112] Zheng, Shuai, Ristovski, Kosta, Farahat, Ahmed, and Gupta, Chetan. “Long Short-Term Memory Network for Remaining Useful Life estimation”. In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. June 2017, pp. 88–95. DOI: 10.1109/ICPHM.2017.7998311.

[113] Yuan, Mei, Wu, Yuting, and Lin, Li. “Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network”. In: *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*. Oct. 2016, pp. 135–140. DOI: 10.1109/AUS.2016.7748035.

[114] Zhang, Chong, Lim, Pin, Qin, A. K., and Tan, Kay Chen. “Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (Oct. 2017). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 2306–2318. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2016.2582798.

[115] Bektas, Oguz, Jones, Jeffrey A., Sankararaman, Shankar, Roychoudhury, Indranil, and Goebel, Kai. “Reconstructing secondary test database from PHM08 challenge data set”. en. In: *Data in Brief* 21 (Dec. 2018), pp. 2464–2469. ISSN: 2352-3409. DOI: 10.1016/j.dib.2018.11.085. URL: https://www.sciencedirect.com/science/article/pii/S2352340918314781 (visited on 08/17/2021).

# Appendix - Contents

# Appendix A

# Python Packages Used

- Tensorflow
- Keras Tuner
- Scikit-Learn
- Matplotlib
- Seaborn

TRITA-EECS-EX-2022:70