POLITECNICO DI TORINO

MASTER's Degree in DATA SCIENCE AND ENGINEERING

MASTER's Degree Thesis

Application of Artificial Intelligence for Data-Driven Prognostic of Finding Remaining Useful Life in Filters

Supervisors

Prof. Danilo GIORDANO

Candidate

Hajali BAYRAMOV

Company Supervisors

Dott. Michele DI FLORIO

April 2023

Abstract

Industry 4.0 has become an established reality thanks to recent technological developments. In this context, Reply Concept, a company where I was employed for a short time to write this thesis, is carrying out a project to build a digital twin for a chemical plant in Sicily, Italy.

The main purpose is to establish a link between the production environment and digital information by creating technological models that analyze the system and connect the real and virtual worlds. The aim of the thesis work is to develop an Artificial Intelligence pipeline for the management and predictive maintenance of the chemical filters, analyzing the data collected by sensors (e.g. mixing pumps, flow meters and pressure sensors) are analyzed.

The importance of determining the Remaining Useful Life (RUL) of filters lies primarily in the capability to avert the failure of equipment that the filters protect. If a filter becomes clogged and is not replaced in a timely manner, it can lead to a decline in equipment performance and even complete failure. By predicting the RUL of filters, maintenance teams can plan ahead for filter replacement, avoiding unexpected downtime and reducing maintenance costs.

The research will analyze the results of various machine learning techniques for predicting the RUL of filters using data-driven prediction. Deep neural networks with specific feature extraction layers, namely Long Short-Term Memory (LSTM) and Implicit Neural Networks (SIREN), are used with fully connected layers for prediction. Moreover, discrete features are represented by continuous values to increase robustness. During training, the data is injected using the sliding window method.

As a case study, I used a publicly available dataset created by Ömer Faruk Eker [1] in a realistic experimental rig. The test rig consisted of a pump, a dampener, particles, flow rate and pressure sensors, and a filter. Since its creation, thorough research and public challenges have been conducted to investigate accelerated clogging phenomena and the prediction of RUL. The dataset can be divided into 3 different operational profiles (i.e. the different sizes of the particles) that make it effective for building applicable predictive models.

State-of-the-art performance metrics have been used by comparing the RUL diverse models to evaluate the methodology. Results show that using the mentioned neural network architectures achieves high-performance results without having much domain expertise. Quantitatively, test results showed less than 20% mean absolute percentage error as well as more than 90% coefficient of determination which is considered good forecasting [2]. Comparing different setups also yielded similar results, which shows the model is robust.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Professor Danilo Giordano who has been very kind and resourceful. He has always been responsive and there, even for my stupidest questions. It has been a great pleasure working under his academic supervision.

I am also grateful to my supervisors and colleagues from Concept Reply, starting from Gerry and Fabio for their support during this long endeavour. Thanks also to Michele for being my supervisor in the company.

Special thanks to my love, without whom I would have been lost and never found motivation. Thanks to my mom, who is always supportive and with full of never-ending love for me. I also want to share my deepest love, respect, and yearning for my dad. You had always been my first supporter till the end. You saw great potential in me and believed in me even more than I do. Thanks, "papa"!

Lastly, I would like to thank my friends and colleagues who made my day when I was down.

"Before we work on artificial intelligence why don't we do something about natural stupidity?"
– Steve Polyak

Table of Contents

List of Tables

List of Figures

Acronyms

PHM
Prognostic and Health Management

RUL
Remain Useful Life

CBM
Condition-Based Maintenance

PbM
Physics-based Maintenance

DDM
Data-driven Model

SOM
Self-Organizing Maps

KbM
Knowledge-based models

IoT
Internet of Things

ARIMA
Autoregressive Integrated Moving Average

AI
Artificial Intelligence

ML

Machine Learning

ANN

Artificial Neural Networks

FC

Fully Connected (Layer)

NN

Neural Networks

DNN

Deep Neural Networks

CNN

Convolutional Neural Networks

RNN

Recurrent Neural Networks

DL

Deep Learning

GAN

Generative Adversarial Networks

LSTM

Long Short-Term Memory

INR

Implicit Neural Representations

SIREN

Sinusoidal Representation Networks

ReLU

Rectified Linear Unit

MSE

Mean Squared Error

MAE

Mean Absolute Error

MAPE

Mean Absolute Percentage Error

R2

Coefficient of Determination

CPU

Central Processing Unit

GPU

Graphics Processing Unit

RAM

Random Access Memory

Chapter 1

Introduction

1.1 Importance of Predictive Maintenance

Individuals who have had access to information have always been able to obtain benefits throughout history, as exemplified by Hippocrates from ancient Greece, who utilized data analysis to study disease patterns and create treatments for his patients. By gathering information on the symptoms, progression, and consequences of different illnesses, he was able to establish the fundamental principles of modern medicine.

The crucial importance of data and analysis in modern times cannot be overemphasized. For instance, data analysis is crucial in making informed decisions in the business world, including marketing, sales, and operational activities. Similarly, data analysis is instrumental in the healthcare sector, enabling healthcare practitioners to enhance patient outcomes and reduce costs. In education, data analysis facilitates the improvement of student outcomes and informs policy decisions. In the public sector, data analysis is utilized to inform policy decisions and enhance public services. In science and research, data analysis is employed to advance knowledge and stimulate innovation. Through analyzing data related to natural phenomena, biological processes, and other relevant factors, researchers can devise new theories, test hypotheses, and achieve breakthrough discoveries.

Data analysis is pivotal, especially in one of the most essential sectors of the contemporary world: **industry**.

- Predictive maintenance: Through the analysis of equipment performance data, firms can anticipate maintenance requirements, minimize costly downtime, and enhance efficiency. This approach helps to curtail expenses and streamline operations.

- Process optimization: The collection and analysis of manufacturing process

data enable organizations to recognize areas for improvement and inefficiencies that may exist within their operations. This information is then utilized to optimize the processes, mitigate waste and enhance the quality of the products.

- Supply chain management: The analysis of data about the performance of the supply chain is an effective approach that firms can utilize to recognize bottlenecks, optimize inventory levels, and improve logistics. By leveraging this information, organizations can reduce expenses and elevate customer satisfaction levels.

- Risk management: Through the analysis of data related to safety incidents, equipment failures, and other risk factors, companies can recognize potential hazards and implement measures to mitigate them. This proactive approach serves to enhance safety, curtail liability, and safeguard the reputation of the organization.

Recent improvements in technologies made way for huge developments in production speed and quantity in all sectors of the industry. Having unexpected pauses in the process is becoming more and more costly each day. Affluent sectors, such as the oil and gas industry can lose $25 M/Day to an interrupt in the platform [3] while the cost can be $1.3 M/Hour for the auto manufacturing industry [4]. These troubles make businesses put more emphasis on predictive maintenance.

Predictive maintenance gained a lot of investments when the third industrial revolution started with the automatization of processes where a huge amount of data is collected. After the empowerment gained by computers use cases of predictive maintenance by smart technologies gained a fast pace. Increased applicability and accuracy of models profit industry, so that research and support for innovation of those technologies also gained a lot of investments [5]. Overall, it is a growing sector which attracts more and more businesses every day.

1.2 AI's Role

Humans are also well known for their desire to imitate the natural world in their creations. We wanted to fly, so we invented planes that fly like birds, and we devised submarines that swim like deep-water fish. Playing god culminated when humans decided to create themselves, a machine with an artificial intelligence close to ours, AI. In the 1940s creation of the programmable digital computer, *"a machine based on the abstract essence of mathematical reasoning"* started the process. Today it reached a point where an AI is capable of drawing very elaborate pictures from the abstract description, writing a piece of software with comments, explaining 'memes' from the internet, and so much more [6, 7].

Figure 1.1: A high-quality photo of a dog playing in a green field next to a lake, totally artificially generated by DALL-E-2 [6].

Although rapid developments make a lot of fear for future advancements, it is undeniable that AI has huge benefits, especially in industry. Applicability of AI on predictive maintenance led to the development of more sophisticated methods which in return benefited the companies. With more advanced methods, businesses can gain more precision in forecasting maintenance with no hindrance.

As mentioned before, the radius of the applicability of predictive maintenance is quite wide. It can be a prediction of the time when an aircraft engine stops working efficiently or of the corrosion in the pipes of an oil refinery to prevent a halt in the process. Day by day AI is gaining more space in this context due to the reasons mentioned in the previous paragraph. There is one real case where a chemical plant in Italy wanted an AI model which predicts the clogging of a filter in advance.

1.3 The Company

I was employed by Concept Reply, a company which is a part of the Reply Group S.p.A. for a short period in a digital twin project where a customer – a chemical plant in Sicily, Italy needed a digital solution to predict the clogging of a filter in advance.

The company is a technology partner that focuses on IoT innovation through hardware and software development. Their expertise lies in providing solutions for Smart Infrastructure, Industrial IoT, and Connected Vehicles. They offer services that cover the entire development process, from the initial idea to implementation,

operation, and support. Concept Reply has a team of IoT specialists who are skilled in hardware design and development, software implementation in embedded systems, as well as edge computing and cloud application development.

1.4 Objectives, Contribution, and Structure

The main scope of the thesis is to research an AI methodology for predicting the clogging of a filter that can be applied in a real scenario. After providing a detailed background check for Predictive Maintenance, state-of-the-art methodologies are investigated. Then the details about publicly available suitable dataset are mentioned in the thesis, after which chosen methodology is explained. A preliminary analysis of the dataset is provided in terms of graphs in the thesis work.

The performance of AI models in terms of forecasting an unknown future is to be tested in this thesis work. Especially, applications of Artificial Neural Networks, a sophisticated state-of-the-art branch of AI are tried. The baseline for the models is set and results from the test phase are present.

Another objective of the thesis is to test the developed models' robustness and sensitivity. Datasets with different characteristics, such as the size of the dataset or the configuration of available data are used to see the sensitivity of the model.

The thesis is structured as follows:

Chapter 2 describes the context and overviews the technologies that enabled Prognostic and Health Management (PHM). Moreover, a detailed literature review of the state-of-the-art methodologies and their background is also presented.

Chapter 3 discusses the details of the publicly available dataset that is used for the experiments. Analysis of the dataset is presented by means of graphs and images. Furthermore, this chapter overviews the techniques utilized for preprocessing.

Chapter 4 describes the data-driven methodologies used in this thesis. It also describes the performance metrics and setup on which the experiments are undergone.

Chapter 5 shows the results from all of the experiments done in the thesis. It contains rich visuals and tables to understand the results better.

Chapter 6, being the last chapter of the thesis, provides the overall conclusions and future improvements.

Chapter 2

Background and Literature Review

In this chapter related background information is provided with the literature review of the state-of-the-art methodologies.

2.1 Industry 4.0

Industry 4.0 refers to the fourth industrial revolution, characterized by the integration of advanced technologies such as artificial intelligence, the Internet of Things (IoT), and robotics into the manufacturing sector. The concept of Industry 4.0 was first introduced in 2011 at the Hanover Fair in Germany [8], and since then, it has gained traction as a key driver of digital transformation in the manufacturing industry.

The first industrial revolution took place in the late 18th and early 19th centuries and was characterized by the use of steam power and mechanical production techniques. The second industrial revolution, which occurred in the late 19th and early 20th centuries, was driven by the widespread use of electricity, the assembly line, and the introduction of the telephone and telegraph. The third industrial revolution, also known as the digital revolution, took place from the late 20th century to the present day and was characterized by the widespread use of computers and the Internet. Industry 4.0 takes the digital revolution a step further by incorporating technologies such as artificial intelligence and machine learning, which allow machines to make decisions on their own and operate autonomously. This leads to the creation of smart factories, where machines are connected and communicate with each other, enabling real-time data exchange and analysis. This leads to increased efficiency, reduced downtime, and improved product quality [10].

The implementation of Industry 4.0 also brings about new business models, such

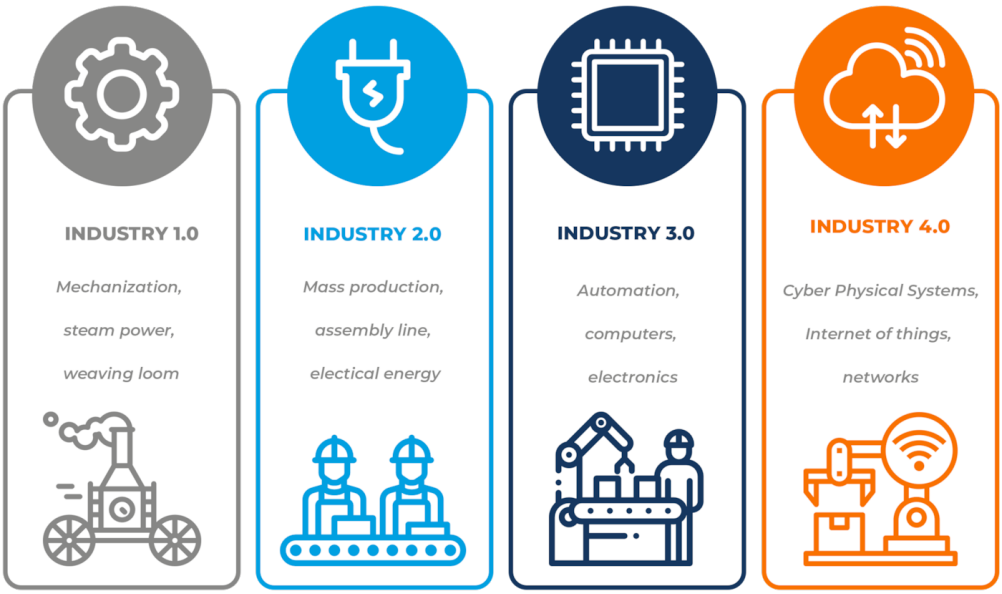

Figure 2.1: Industrial revolution steps in short [9]

as the use of cloud computing and the emergence of Industry 4.0 service providers. This allows companies to access advanced technology and expertise without having to invest in expensive equipment and resources.

Industry 4.0 is a major step forward in the evolution of the manufacturing industry and has the potential to revolutionize the way products are produced and services are delivered. While the implementation of Industry 4.0 brings about new challenges and requires significant investment, it has the potential to drive economic growth and improve the lives of people around the world.

Enabling technologies [11]

- **IoT**. The Internet of Things (IoT) is a network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity which enables these objects to connect and exchange data. IoT devices can be connected to the Internet and other connected devices, allowing for the collection and exchange of data to improve efficiency, and accuracy, and enhance user experiences. The IoT concept has the potential to revolutionize the way we live, work and interact with our environment by creating smart homes, smart cities, and a connected world. The specific branch of IoT adapted to Industry 4.0 is called "industrial Internet".

- **Big Data/Industrial Analytics**. Big Data refers to the large and complex

data sets that are generated from various sources and are too big, fast, or complex for traditional data processing systems to handle. The term "Big Data" encompasses a wide range of data types including structured, semi-structured, and unstructured data. With the growth of the Internet, social media, and IoT devices, organizations are now collecting and storing vast amounts of data.

Industrial Analytics refers to the use of big data and advanced analytics techniques to optimize industrial processes and make better-informed decisions. This involves analyzing vast amounts of data generated by industrial processes, equipment, and systems to gain new insights, improve efficiency, and make data-driven decisions. The goal of industrial analytics is to turn big data into actionable information that can be used to improve operations and drive business outcomes.

- **Cloud Manufacturing**. Cloud Manufacturing refers to the use of cloud computing technology to manage and optimize manufacturing processes. It involves using cloud-based tools and platforms to store, process, and analyze large amounts of data generated by manufacturing operations. The goal of cloud manufacturing is to improve efficiency, reduce costs, and enhance the overall competitiveness of the manufacturing industry.

  Cloud manufacturing enables manufacturers to access their data and systems from anywhere, at any time, using any device with an Internet connection. This allows them to make real-time decisions based on up-to-date information and reduces the need for expensive on-premise IT infrastructure. Additionally, cloud manufacturing enables manufacturers to collaborate more effectively with suppliers, customers, and other stakeholders, improving supply chain management and increasing agility.

- **Machine Learning (ML) and Artificial Intelligence (AI)**. Machine learning involves training algorithms to learn patterns and relationships in data, allowing them to make predictions and decisions based on new data inputs. In Industry 4.0, machine learning algorithms are used in a variety of applications such as predictive maintenance, quality control, and supply chain optimization. For example, ML algorithms can analyze large amounts of production data to identify potential equipment failures before they occur, reducing downtime and maintenance costs.

  AI, on the other hand, refers to the development of computer systems that can perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, and solving problems. In Industry 4.0, AI is used to automate and optimize manufacturing processes, resulting in increased efficiency and productivity. For example, AI-powered robots can be used to

perform repetitive tasks, freeing up workers to focus on more complex and creative tasks.

ML itself is divided into 3 main categories:

- **Supervised** learning is a form of machine learning that deals with problems where data has already been labelled. This means that each data point includes features and a related label. Supervised learning algorithms aim to learn a function that can map the features to the corresponding label, based on input-output pairs from the data [12]. The algorithm uses this labelled training data, consisting of a set of examples, to create an inferred function [13]. Each example consists of an input object and a desired output value, known as the supervisory signal. By analyzing the training data, the algorithm creates a function that can be used to predict the label for new examples. To be successful, the algorithm must be able to accurately determine the class labels for unseen instances, which requires it to generalize from the training data in a sensible manner. The ability of the algorithm to do this is measured by the generalization error.

- **Unsupervised** learning is a machine learning technique that handles situations where there is no labelling information available for the data. The objective of unsupervised learning is to identify patterns or relationships in the data without being given a specific outcome to predict. The algorithm is given a set of data and must independently uncover the structure within it.

- **Reinforcement** learning involves training an agent to make decisions by experiencing the consequences of its actions in an environment. The agent is rewarded or punished based on its decisions and gradually learns to choose actions that maximize its reward.

Over the decades, many ML algorithms have been developed for various purposes. Well-conducted surveys ([14, 15, 16]) are present in the literature. For example, [14] talks about state-of-the-art techniques, along with real-world applications and research directions very thoroughly. Future prospects have been analyzed in [15].

A detailed review is dedicated to Deep Learning (DL) which is *"a part of a broader family of ML methods based on artificial neural networks (ANN) with representation learning [17]"* in upcoming pages of this section.

2.2 Digital Twin

A digital twin is a virtual representation of a physical object or system, created using real-time data and computer simulations. It is a digital replica of a physical product,

process, or service, and is used to analyze, design, and optimize various aspects of the real-world counterpart. Digital twins are becoming increasingly popular across various industries, such as manufacturing, healthcare, and transportation, due to their ability to improve operational efficiency, reduce costs, and enhance customer experiences.

The concept of a digital twin was first coined in 2002 by Dr Michael Grieves at the University of Michigan [18]. It has since evolved into a powerful tool for businesses to gain insight into their operations, optimize processes, and make data-driven decisions. Digital twins use sensors and other sources of data to create a virtual model of a physical object or system, which is then used to simulate and analyze its behaviour. This allows companies to identify potential problems, test solutions, and make changes to improve the performance of the physical system.

The benefits of digital twins have been analyzed thoroughly in research conducted by [19]. One of the main benefits of digital twins is the ability to improve operational efficiency. By creating a virtual representation of a system, businesses can optimize processes and reduce downtime, resulting in increased productivity and reduced costs. For example, in the manufacturing industry, digital twins can be used to optimize production processes, reduce waste, and improve product quality. This can result in increased efficiency, lower production costs, and a better end-product for customers.

Another benefit of digital twins is their ability to enhance customer experiences. By using digital twins, companies can gather data on customer behaviour and preferences, and use this information to create personalized experiences. For example, in the healthcare industry, digital twins can be used to create individualized treatment plans for patients, based on their unique needs and conditions. This leads to better outcomes and improved patient satisfaction.

Digital twins also have the potential to revolutionize the way businesses approach maintenance and repairs. By having a digital representation of a physical system, businesses can use predictive analytics to identify potential problems before they occur, reducing the likelihood of unexpected downtime. This allows companies to take proactive steps to address problems, resulting in improved operational efficiency and reduced costs.

The use of digital twins is also becoming increasingly popular in the field of the Internet of Things (IoT). IoT devices and sensors can be used to collect real-time data, which is then fed into the digital twin. This allows companies to analyze the behaviour of their physical systems in real-time, and make data-driven decisions to optimize performance.

Despite its many benefits, the use of digital twins is not without challenges. One of the main challenges is data privacy and security. As digital twins rely on large amounts of data, it is important to ensure that this data is protected from cyber threats and unauthorized access. Another challenge is the need for specialized skills

and knowledge to create and maintain digital twins. Companies must invest in the necessary resources, such as specialized software and trained personnel, in order to effectively use digital twins. Review done by [20] found that multidisciplinarity, standardization, and global advancements are the main challenges to open research.

In conclusion, digital twins are a powerful tool that can revolutionize the way businesses operate. By creating a virtual representation of a physical system, businesses can optimize processes, reduce costs, enhance customer experiences, and improve operational efficiency. However, the use of digital twins is not without challenges, and companies must be aware of the potential risks and invest in the necessary resources to effectively utilize this technology. As the use of digital twins continues to grow, it is likely that they will become an essential tool for businesses across various industries, helping to drive innovation and improve outcomes.

2.3 Maintenance

There are 2 main categories in maintenance philosophies [1]:

1. Reactive Maintenance
2. Proactive maintenance

Reactive maintenance is usually done when the issue is noticed or a breakdown happens. No planning is done for reactive maintenance and usually, the problem halts the whole process or the part that needs care becomes useless or inefficient. Chain reaction causes ineffective work processes and in huge applications, big amounts of money are lost. This form of maintenance is the oldest type of maintenance and is gradually vanishing in large-scale applications. Nevertheless, it is still widely used in basic components such as valves at home where there is little to no risk and/or unnecessary maintenance time and the cost is needed.

Proactive maintenance, on the other hand, is planned beforehand. Starting from the last mid-century proactive maintenance has gained preference over reactive maintenance, which is done after the need occurs [21]. It became apparent when industrial advancement brought upon frequent equipment wear out which is intolerable as it requires more downtime and labour cost. 2014 Gartner report [22] cites that downtime costs can be $300.000 per hour on average, sometimes reaching as high as half a million. As the industry goes forward, this cost is also multiplying. So companies are eager to invest more and more not to halt the process.

One way of doing proactive maintenance is to schedule a timetable for when the care is needed, which is called preventative maintenance. Maintenance is done in a fixed period of time without any feedback from the system. Although it is a more effective way than reactive, it has still some complications, such as a waste of time and resources in case of the absence of fault.

The mere idea of having resources dedicated to unnecessary maintenance led to the development of predictive maintenance. The main trait that distinguishes it from its predecessor is having the ability to determine the scheduling time based on system needs. As it is a highly complex and costly task, predictive maintenance is only limited to critical and technically possible tasks [23]. A comprehensive survey has been conducted by Ran et al. [24] with emphasis on system architectures, purposes and approaches.

Condition-Based Maintenance (CBM) is a type of predictive maintenance in which maintenance activities are performed based on the actual condition of the equipment. The equipment's health is constantly monitored and when it reaches a predetermined point of degradation, maintenance is triggered to prevent failure [25, 26].

There are two main branches of CBM: Diagnostics and prognostics. Diagnostics in the context of CBM involves identifying and reporting any anomalies in signals, determining the cause of the issue, and evaluating the current state of the asset [1]. With diagnostics, the goal of CBM is to halt and schedule maintenance once an abnormality is detected, so as to prevent the system from failing. If degradation is discovered, maintenance must be performed immediately to avoid potential failures.

Prognostics, on the other hand, alongside being a relatively new and complex task gives the opportunity to prepare for the actual maintenance. Because while preparation is ongoing for the maintenance system is up and running, only downtime is calculated when the actual maintenance is performed. Thus, decreasing the overall time compared to the former methodology.

After completing the primary steps, such as health assessment, severity detection, and degradation detection comes the second phase of prognostics, referred to as "true prognostics," which focuses on predicting the exact time of failure by projecting the trend of degradation and calculating the remaining useful life (RUL). This phase is characterized by techniques such as time series analysis, extrapolation, propagation, trending, projection, and tracking [1].

In literature, prognostic approaches are divided into 4 main subcategories:

- **Physics-based models**, also known as PbM approaches, use engineering and scientific knowledge [27] to evaluate the health of a system through a set of equations, either for diagnosis or prognosis. The key advantage of PbMs lies in their ability to predict long-term behaviour through degradation models [28]. Luo et al. [29] proposed a generic process for developing prognostics using PbMs, which involves creating a model of the system and its degradation, where fast dynamic variables describe the behaviour of the system and slow dynamic variables describe its degradation. The appropriate scenario (feature estimation) and RUL are estimated by simulating various random scenarios and comparing the results to actual data.

A review conducted by Cubillo et al. [30] identifies the potential PbM for prognostics in rotating machinery. A very interesting article written by Aivaliotis et al. [31] proposes to employ the Digital Twin approach to keep track of a robot's health status and align the simulated behaviour with the actual behaviour of the robot. The results of the simulation can provide insight into the future behaviour of the robot, predict the quality of the products to be produced, and calculate the robot's RUL. The proposed method is tested in a case study from the white goods industry to determine if the robot is likely to experience a failure in the next 18 months.

- **Data-driven models** (DDMs) use data collected from condition monitoring or historical events instead of relying on system physics or expert knowledge. They track the asset's degradation through techniques such as regression, exponential smoothing, and neural networks, or by identifying similar patterns in the data to estimate the RUL[32]. These models rely on past degradation patterns to predict future degradation and do not consider system inputs or operational profiles. Since they don't take into account specific information about the asset, they are considered a black-box operation [33]. DDMs can be divided into two categories: statistical models and AI models.

  The current leading models in statistical prognostics can change based on the industry and the specific situation, however, some popular models include:

  1. Regression models - These models use past data to create a connection between an asset's degradation and one or more input factors, which then allows them to calculate the asset's RUL [34, 35, 36].
  2. Time series models - These models analyze time-series data, such as data collected over a certain period from a single asset, to identify patterns and trends in degradation that can be used to predict the RUL [37, 38].
  3. Survival analysis models - These models are commonly utilized in reliability engineering and consider the time-to-failure of an asset, taking into account censored data (for example, an asset that has not yet failed but for which data is available up to a certain point in time).
  4. Weibull models - These models are widely used in reliability engineering and assume that the time-to-failure of an asset follows a Weibull distribution.
  5. Proportional hazard models - These models are often employed in survival analysis and assume that the hazard (or risk) of failure is proportional to the asset's degradation over time.

  Most recent advancements in statistical DDMs and their applications are well reviewed in the following articles ([37, 38, 34, 26, 39, 40])

AI or machine learning models aim to detect intricate patterns and make informed decisions based on empirical data. These models are suitable in situations where solutions to problems are hard to specify, but there is enough data or observations available. Some of the common machine learning methods used for supporting detection, diagnostics, and prediction include Artificial Neural Networks (ANN), Self-Organizing Maps (SOM), and decision trees [41, 42, 43]. A detailed review of AI models that are used in prognostics is done in the following sections of this chapter.

- **Knowledge-based models** (KbM) use prior expertise and understanding of a system to predict its future performance. These models are a form of model-based approach in prognostics and are built using mathematical equations and physical laws. The models take into account factors such as the system's design, operating conditions, and environmental factors to provide a prediction of the system's remaining useful life. The goal of these models is to offer a more accurate and trustworthy prediction compared to data-driven methods. They are widely used in various industries, including aerospace, defence, energy, and manufacturing.

  The simplest method for performing prognostics is through knowledge or experience-based approaches, where past failures of systems are analyzed statistically to forecast the RUL of the system. These models are a type of automated solution that mimics the decision-making process of a human expert in a specific field [32].

- **Hybrid models** is a combination of knowledge-based and data-driven approaches, utilizing the strengths of both methods to achieve a more accurate and reliable prediction of the RUL of a system. This approach merges human expertise with the power of data analysis, allowing for improved predictions compared to either approach alone. Hybrid models are useful in situations where there is limited knowledge or insufficient historical data and can include expert systems incorporating data-driven models or data-driven models that incorporate expert knowledge through prior probabilities or rules.

  Eker [1] has conducted thorough research on hybrid prognostics and its application to well-controlled engineering systems.

2.4 Deep Learning

As the complexity of the tasks and dimension of raw information increased, simple ML models started not to meet the requirements. Shallow extraction of features becomes difficult and more sophisticated – deeper multi-level architectures started

to arise, where the name of Neural Networks (NN) are at the heart of it. Complicated tasks, such as Natural Language Processing, Computer Vision, or Anomaly Detection leaned more towards DL as it generates more accurate results while intelligently analysing the data on a large scale. By tuning the hyper-parameters in the model information well hidden in different layers can be extracted.

DL history can be traced down to Walter Pitts and Warren McCulloch when they developed the first model inspired by neural networks in the human brain in 1943 [44]. Since then DL community has been introducing new concepts, such as back-propagation [45], Convolutional Neural Networks (CNN) [46], long short-term memory (LSTM) [47], Generative Adversarial Neural Networks (GAN) [48] etc. In 2009 14 million labelled images are introduced by Deng et al. under the name of ImageNet [49].

2.4.1 Neural Networks

Artificial neural networks (ANNs) are computational ML models based on the structure and function of biological neural networks in the human brain. An ANN consists of multiple interconnected processing nodes organized into layers, also known as artificial neurons. The processed input passes through these layers, with each neuron processing the input and generating output. It is then passed on to the next layer. The connections between neurons are weighted, determining the strength of their influence on each other.

To achieve accurate performance for a specific task, the weights of the network must be optimized through training. This is accomplished by presenting the network with a set of input-output pairs from a training dataset and adjusting the weights to minimize the difference between the network's predictions and the actual outputs. The most commonly used method to train ANNs is through backpropagation [45], using gradient descent [50] to adjust the weights.

However, there are also challenges associated with ANNs, such as overfitting, where the network becomes too complex and fits the training data too closely, leading to poor performance on unseen data. The internal workings of the network can also be difficult to interpret, and training can be slow. Despite these challenges, ANNs have been widely adopted and have been successful in many applications, due to their ability to learn from data and perform well on a wide range of tasks.

2.4.2 NN used in Time Series analysis

As was discussed previously, ML algorithms have been a great tool on hand for the analysis of time series. Before more complicated ML models domain is dominated by mostly statistical methods, such as moving averages and exponential smoothings. More complex applications promoted the development of many AI models and

testing of other preexisting models on time series analysis (e.g. Support Vector Regressors, ensemble methods like Random Forest or XGBoost, etc [51, 52]). Sophisticated PHM objectives, such as identifying RUL, ever-increasing "big" data from IoT devices, and increased computational resources led to the development, improvement, and applications of NN models on time series data. Examples are introduced in the following few paragraphs.

- **Deep Neural Networks (DNN)** consists of many layers of neurons also known as perceptrons connected which is the main part of the network that is inspired by brain circuits [53]. Perceptron is shown in figure 2.2. The activation function is to allow networks to handle complex problems with fewer nodes using the nonlinearity property. Well-known activations functions are ReLU, Tanh, Sigmoid etc [54]. DNNs can overcome many problems that simple perceptrons cannot handle.

  DNNs have many capabilities that make them a very good candidate for the usage of time series forecasting [55]. It is robust to noise, it has nonlinearity and multivariate input properties, and it can do multi-step forecasting. Liu et al. [56] built a reinforcement learning NN model for time series prediction. Another good example is built by Pavlyshenko [57] using Deep Q-Learning applied to sales time series. Ismail Fawaz et al. [58] proposed a very extensive study by training 8730 DL models on 97 time series datasets.

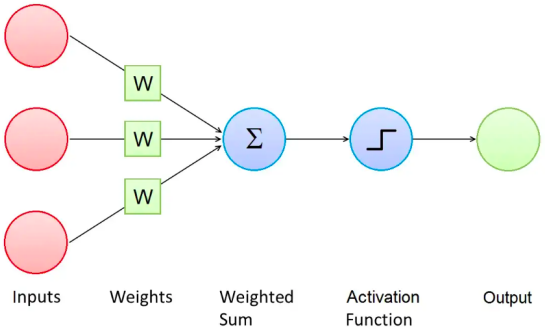

Figure 2.2: Perceptron [59]

- **Convolutional Neural Network (CNN)** can learn from salient features that are otherwise impossible to extract. Although it is generally used in Computer Vision, this trait makes it a suitable choice for many univariate, multivariate, or multi-step time series analyses and forecasting [60]. Wibawa et al. [61] improved the quality of time series analysis by developing a novel methodology of Smoothed-CNN. Application of CNN on RUL forecasting of rolling bearings are studied by Wang et al. [62]. Moreover, Yang et al. [63] proved that the proposed Double-CNN based NN architecture achieves

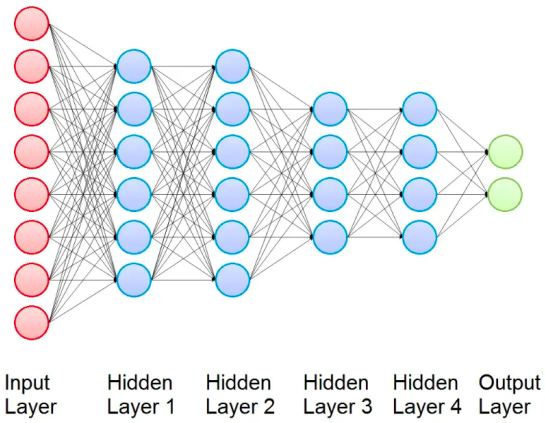

Figure 2.3: DNN [59]

higher prediction accuracy and robustness on RUL prediction compared with state-of-the-art methods.

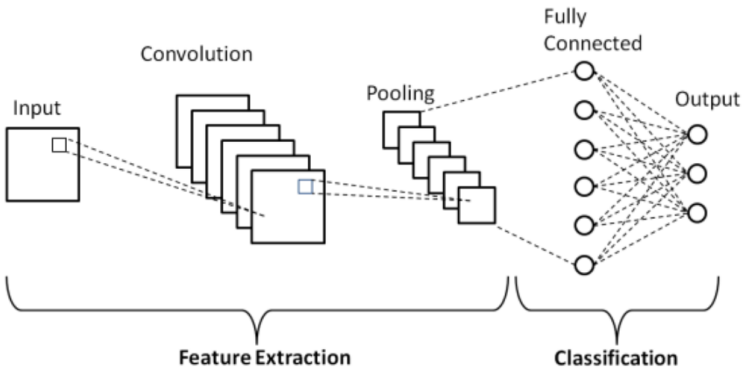

Figure 2.4: Architecture of CNN [64]

- **Recurrent Neural Network (RNN)**, such as the Long Short-Term Memory (LSTM) network [47], offer a unique capability compared to other types of neural networks like fully-connected DNNs and CNNs [55]. Unlike these other types of networks, RNNs are specifically designed to handle input data that comes in the form of sequences, where the order of observations is important. This means that RNNs can capture the temporal relationships between observations, making them well-suited for time series analysis and other problems that involve sequences. Structure of LSTM architecture is well depicted in figure 2.5 by [65]

  These features of RNNs attracted much attention in the PHM community for

building forecasting models, specifically for RUL estimation. Yan et al. [66] modelled an LSTM neural network for the prediction of long-term gear life. Boujamza and Lissane Elhaq [67] built an attention-based LSTM for RUL forecasting of aircraft engines. A similar attention mechanism is used by Li et al. [68] for RUL prediction of aircraft engines with an enhanced CNN-LSTM. A genetic algorithm optimized RNN-LSTM model for RUL estimation of turbofan engine was developed by Tai Chui et al. [69]. Nie et al. [70] achieved a highly accurate model using an integration of ARIMA and LSTM models in the estimation of RUL of water hydraulic high-speed on/off valves. A case study by Huang et al. [71] proved that LSTM neural network can be used to estimate the RUL for systems with abrupt failures.

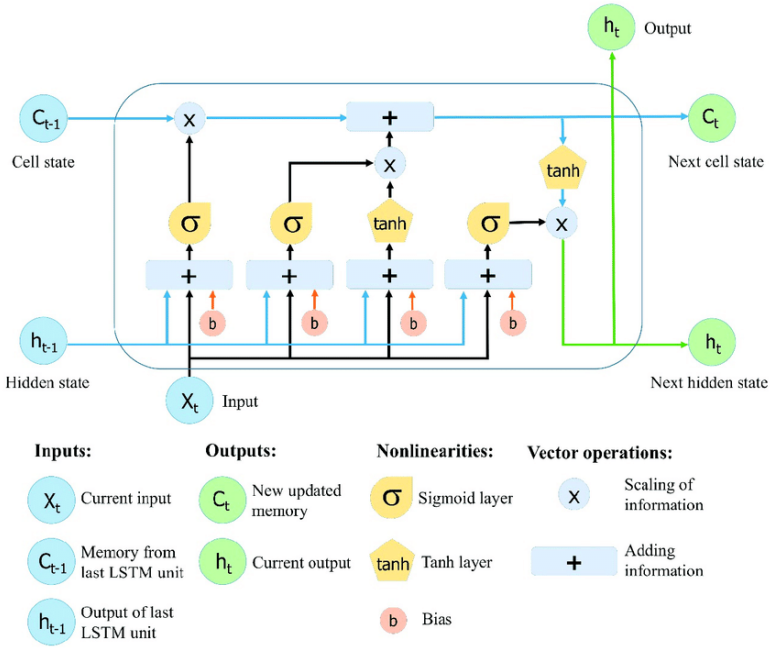

Figure 2.5: Structure of LSTM [65]

- **Implicit Neural Representations (INR)** are the internal representations created within a neural network that is learned through the processing of training data, rather than being explicitly designed by humans. As input data is processed through multiple layers of artificial neurons, the network develops representations that grasp the correlations and connections present in the data.

  Sitzmann et al. [72] proposed a methodology that leverages periodic activation functions for INR and proved that dubbed sinusoidal representation networks (SIRENs) can be ideal for representing complex natural signals, such as time

series. In SIREN sine is used as a periodic activation function.

Chapter 3

Dataset

3.1 Dataset

3.1.1 Source

In 2020 PHM Society announced a data challenge to create a prognostic model for filter clogging based on a dataset created by Eker et al. [1]. An experimental rig was constructed to gather run-to-failure filter clogging data. The simulation rig consisted of a filter, liquid tanks, a stirrer, a peristaltic pump, a pulsation dampener, and sensors to measure pressure and flow rate. In order to simulate filter clogging faster different-sized polyetheretherketone particles are mixed in water in various concentrations.

The main objective is to predict the RUL of the filter, where the $20psi$ difference between upstream and downstream pressure is considered as the threshold. 48 experiments are simulated in $10Hz$ frequency under 3 equal-sized sets of possible particle sizes:

- small - $(45 - 53\mu m)$
- medium - $(53 - 63\mu m)$
- large - $(63 - 75\mu m)$

Every set has equal numbers of 4 various ratios of particles:

- 0.4%
- 0.425%
- 0.45%
- 0.475%

In each experiment information is available as the following set of features:

- Time (s) - time passed since the start of the experiment
- Flow Rate (ml/m) - flow rate of fluid running through experiment pipe
- Upstream Pressure (psi) - the pressure of fluid entering the filter
- Downstream Pressure (psi) - the pressure of the fluid leaving the filter
- Particle Size (μm) - particle size
- Solid Ratio (%) - the ratio of solid particles in the fluid

3.1.2 Preprocessing

Primary analysis and Cleaning

This section analyses experimental data after basic preprocessing steps, such as *Differential Pressure (ΔP)* calculation and frequency drop. The pressure drop between Upstream and Downstream or ΔP is calculated simply by subtracting the latter from the former. Frequency is dropped from 10Hz to 1Hz. Averaging and median techniques are tried where every 10 seconds are aggregated using the appropriate method and the former is selected because of resulting in smoother and stable points, 3.1 and 3.2 respectively.

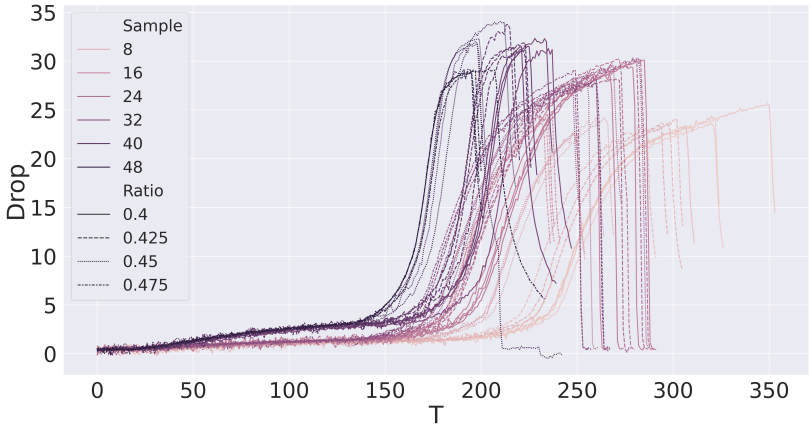

Figure 3.1: ΔP/Time graph after using averaging technique.

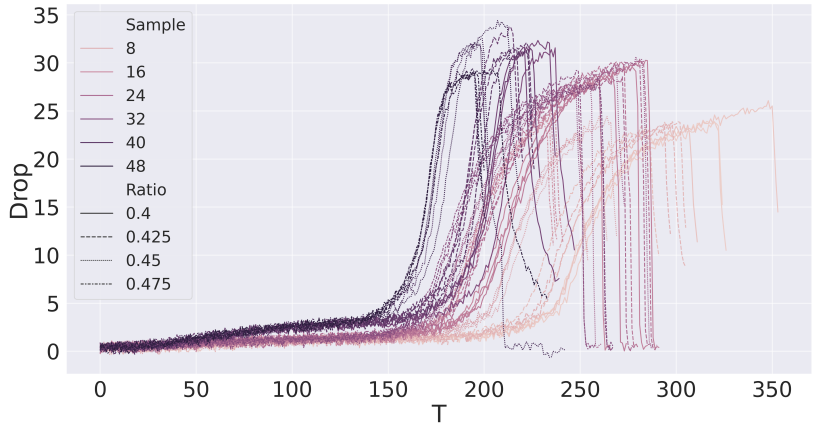

Figure 3.2: ΔP/Time graph after using median technique.

Figure 3.3 shows flow rates of 48 experiments against time clustered by specific colour from the palette based on *Sample* (considering that particles in the first 16 experiments are small, the next 16 are medium, and the rest are large sized). Figure 3.4 depicts the ΔP-Time graph grouped by their *Size*. This way we can see also the difference between different *Ratio*s. These two analyses proved that the larger the *Size* and the bigger the *Ratio* make filter clogging faster. Moreover, the scatter plots depicted the relationship between *Flow* and ΔP in figure 3.5. The last graph hinted us that samples with large particles are a little different than the rest which might affect the model's sensitivity towards that samples. This theory is tested in next sections of my thesis.

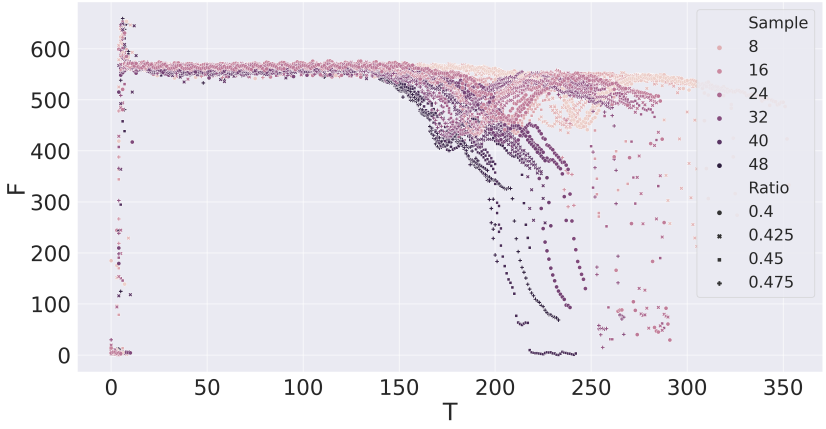

Figure 3.3: Flow Rate/Time graph

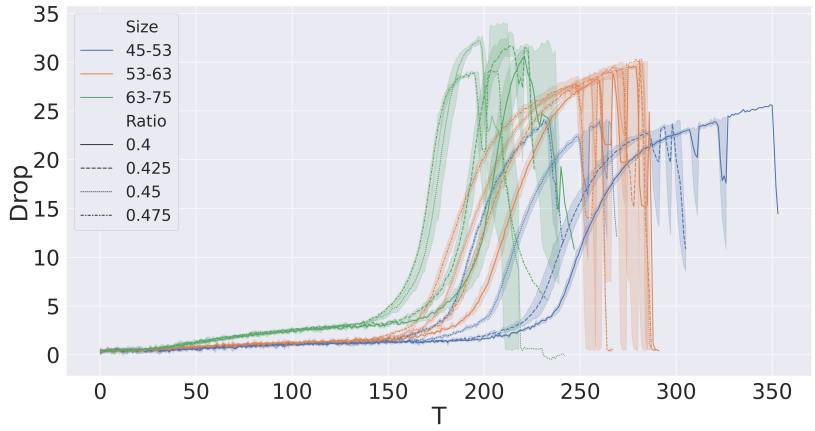

Figure 3.4: ΔP/Time graph grouped by Size

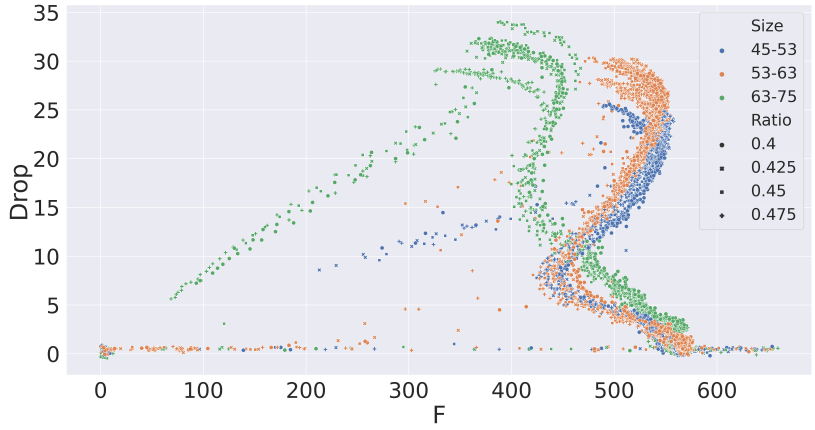

Figure 3.5: ΔP/Flow graph

We can see from the first inspections that data has outliers at the beginning and at the end of each experiment. The former is because simulation data is collected from a physical experimental rig and it takes some time after starting the experiment to reach certain values. On the other hand, we can remove outliers at the end by clipping the data samples after ΔP reaches a certain threshold, which is set as $20psi$. Records after flow reach $550ml/m$ are dropped as well. After the cleaning phase dataset became as follows (figure 3.6, 3.7, 3.8):

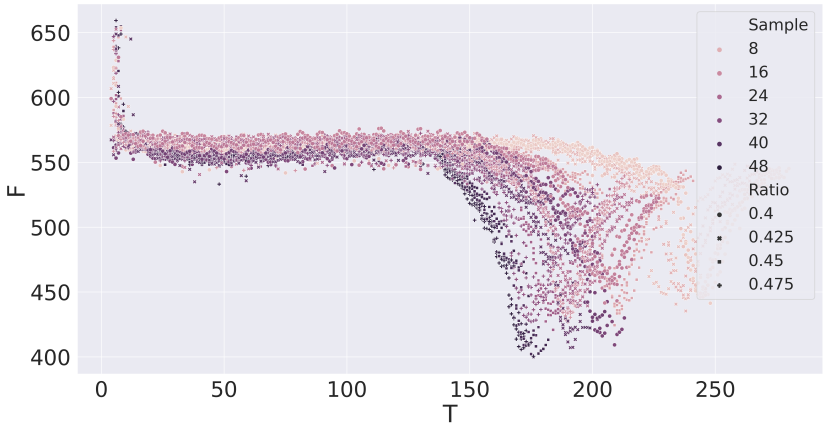

Figure 3.6: Flow Rate/Time graph after cleaning

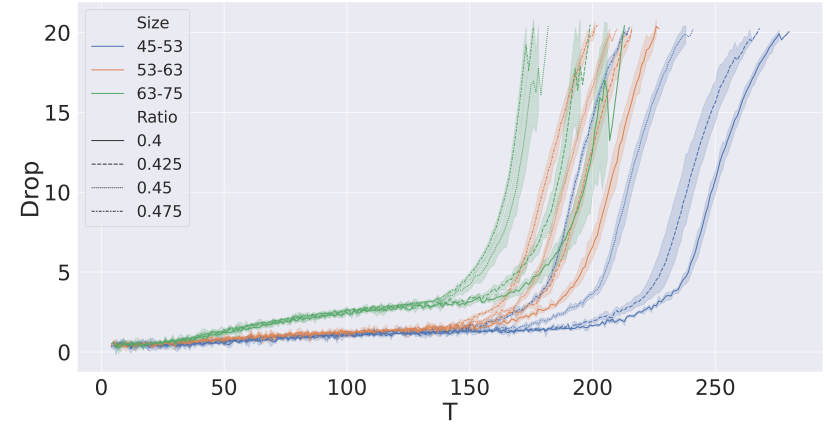

Figure 3.7: ΔP/Time graph after cleaning

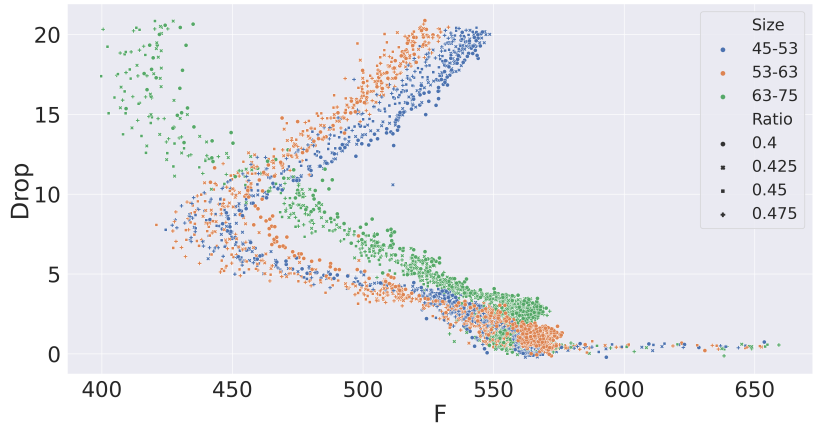

Figure 3.8: ΔP/Flow graph grouped by Size after cleaning

Furthermore, *RUL* feature was generated by taking the reference point as the last sample to have 0 RUL, then subtracting the current timestamp from the last timestamp iteratively (T of the last sample - T of the current sample). The following figures (3.9, 3.10, 3.11) show the relationship between *RUL* and other features. These graphs also show a slight difference between the processes of large-particle-sized samples and the others. Moreover, *Size* is converted to a numerical value by representing its mean value (e.g. 45-53 to 49). The downstream pressure feature is removed since it is represented by Drop.

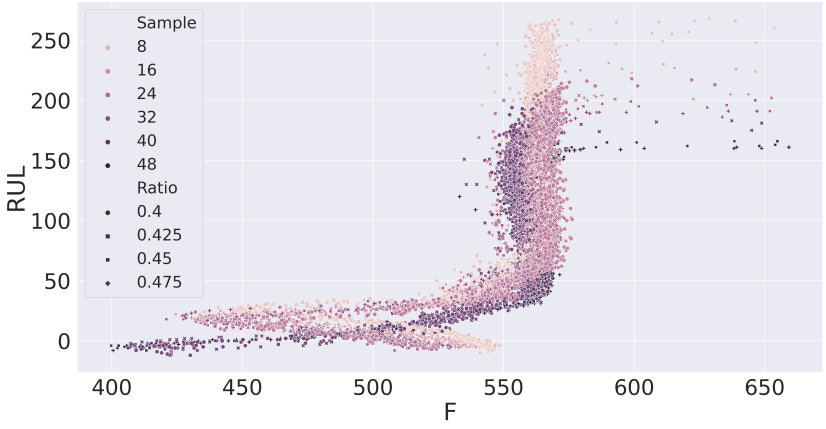

Figure 3.9: RUL/Flow graph

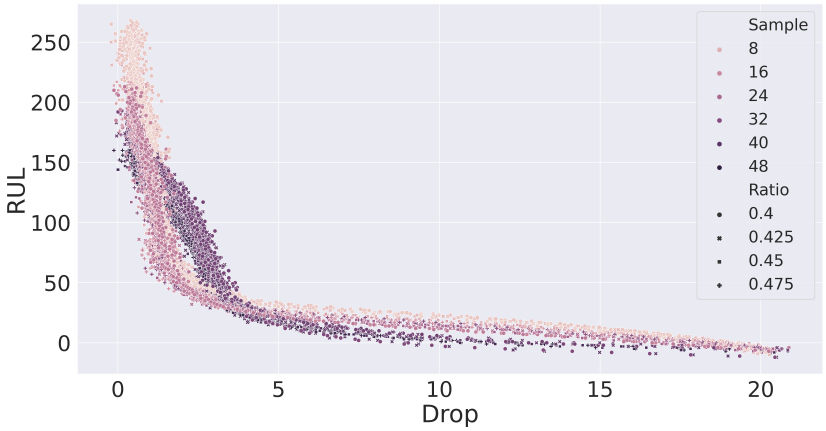

Figure 3.10: RUL/ΔP

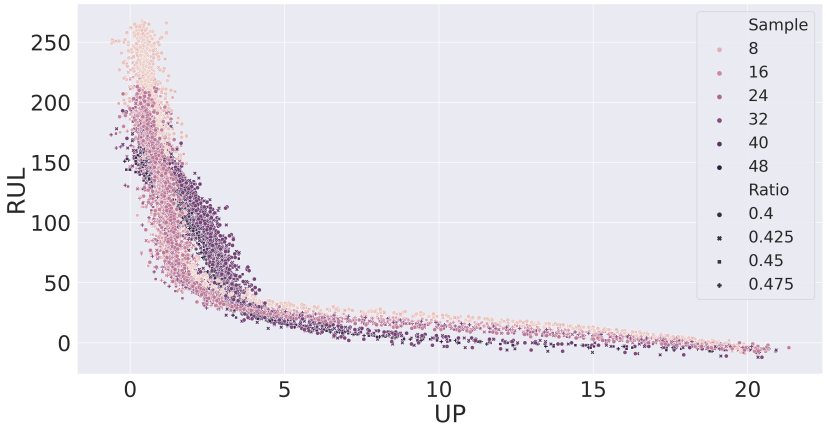

Figure 3.11: RUL/UP graph

Chapter 4

Methodology

4.1 Preliminaries

Normalization

Normalization is a common practice in preprocessing step for datasets. It is the process of scaling the values between the fixed range so that the data can be compared on a common scale. For some scale-sensitive algorithms, like neural networks, it can cause problems. For example, features that can reach very high values can have more weight compared to others that have significantly smaller values while in reality weights of these features should be similar (e.g. *Flow Rate* is fluctuating in a couple of hundreds while *Ratio* can only be a floating point between 0 and 1). Another benefit is to help to prevent overfitting. Overfitting is when the model is adapted too much to the training sample set so that when a new unknown sample is given it is not able to infer correctly. This is because the model is paying too much attention to noise and not to the underlying pattern.

For this dataset, the Min-Max scaler (eq. 4.1) is utilized since the dataset has known minimum and maximum values and it is not normally distributed. Data is normalized between 0 and 1 not only for the continuous features, such as *Flow, Upstream Pressure, and* ΔP but also so-called discrete features which are *Size* and *Ratio*. This is because the goal is to make the model more generalized and robust so that when it comes across something that was not in training or tests it can infer. This is particularly possible because the minimum and maximum values these features can achieve are somewhat known. For instance, *Ratio* cannot be more than 100% and less than 0%.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Minimum maximum values for each set are fixed by considering the training

dataset only and then applying a transformation to the test dataset without changing fixed values.

4.1.1 Sliding windows

The sliding window technique examines and handles data by shifting a fixed-sized window across the input data. This approach breaks down a larger input sequence into smaller, more manageable sections that can be analyzed independently.

To apply the sliding window technique, a window with a fixed size (*window size*) is moved across the input sequence, one position (*stride*) at a time (figure 4.1). Each shift is one batch of data to feed the model as input. Time series analysis is a common application for the sliding window technique.

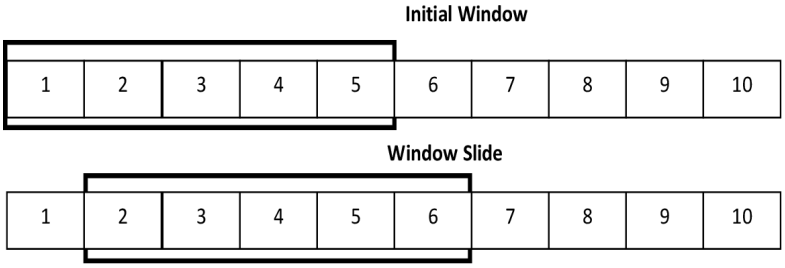

Figure 4.1: Process of sliding window with *stride*=1 and *window size*=5 [73]

Input data is considered as all samples of all window features except *RUL*. On the other hand, the target value is the last value of *RUL*. This way model can be trained to predict one value of *RUL* by taking a sequence of data with a length of *window size*. Figure 4.2 shows an example from preprocessed data.

Input (X)

F	UP	Drop	Size	Ratio
0.8662619495	0.05400540054	0.0307294914	0.45	0.5
0.8660052253	0.05362794344	0.03230307889	0.45	0.5
0.8631812624	0.0496501263	0.03643003474	0.45	0.5
0.8568235706	0.04866293081	0.02986847184	0.45	0.5
0.8548905905	0.05081153277	0.03185772394	0.45	0.5
0.8597683446	0.05249557214	0.02906683293	0.45	0.5
0.8667602957	0.04244940623	0.03064042042	0.45	0.5
0.8687536814	0.05046311083	0.03346100175	0.45	0.5
0.8663676588	0.04906942307	0.0333125501	0.45	0.5
0.865053837	0.04593362562	0.03120453668	0.45	0.5

Target (y)

RUL
0.7

Figure 4.2: Sample Input and Output shown as window

4.2 Architectures

As was discussed in section 2 LSTM models are capable of easily extracting the underlying information from the sequential data. Recent research has proven that this type of model can achieve high accuracy in the applications of time series analysis, specifically prognostic health management systems like RUL forecasting. That is why this architecture was chosen first to experiment.

Secondly, SIREN architecture has promising results from research that show the applicability of the model to time series prediction [72]. According to the research, RUL estimation analysis has not been conducted with mentioned architecture. Thus, as a second model SIREN is chosen.

Both models require the input to have sequential data to extract the latent features hidden inside continuance. After preprocessing the utilized dataset has 3 features that are continuous: *Flow rate, Upstream pressure,* ΔP. The remaining two features do not change during the experimentation time period: *Size, Ratio*. Taking into account this information following two architectures are built.

4.2.1 LSTM

LSTM by itself always requires data to be in some certain length. In the case of this experimentation *window length* is predefined. A general model with LSTM architecture can be visualized as a network consisting of multiple LSTM layers followed by fully connected layers which then generate a single value, prediction of RUL. However, discrete values cannot be fed to the LSTM layer so they are concatenated to the result of the last LSTM layer and then fed to the first fully connected layers (figure 4.3).

LSTM networks are mostly consisting of redundant neurons, meaning that training on these types of networks can end up “overfit". Common practice is to use dropout for these networks. Dropout is used to prevent this. It is a regularization technique that makes some random neurons be ignored during training so that their weights are not updated. The most common activation function, ReLU is used in this LSTM model [74].

4.2.2 SIREN

A very similar network is built with the only difference in the continuous feature extractor where LSTM layers are exchanged with SIREN. Both models are tested on separate sets of settings meaning that they do not share any hyperparameters in common. A Sine activation function is used as the original paper suggested [72].

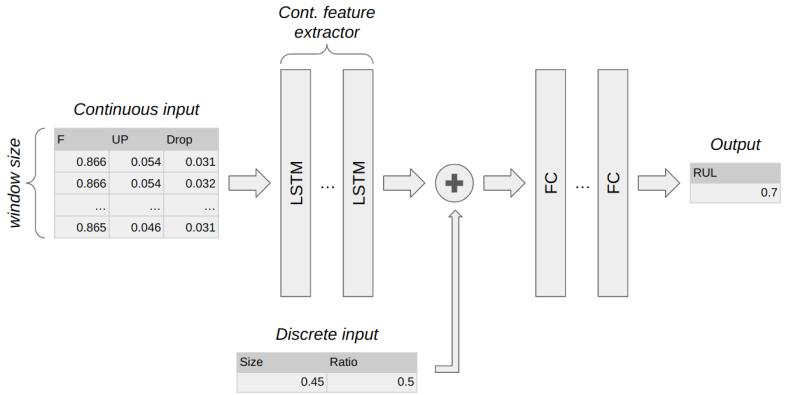

Figure 4.3: General model of LSTM built for experiments

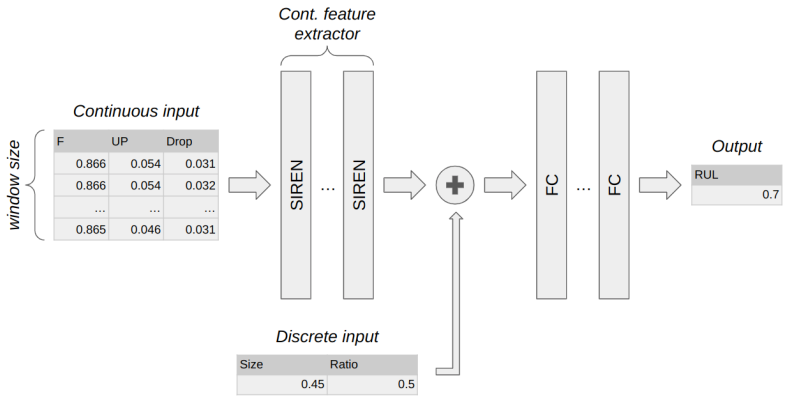

Figure 4.4: General model of SIREN built for experiments

4.2.3 Model fitness

Measuring the difference between actual and predicted values not only by error but also by accuracy equations is quite common in the field. Accuracy is the measure that shows, as the name itself suggests, how accurate the trained model is.

Even though it is not "accuracy" by itself, R2 or coefficient of determination is a good tool to measure how well the regression line fits the data points [75]. What makes it an especially good metric for evaluation is its range. The R2 value ranges from 0 to 1, with 1 indicating a perfect fit of the model to the data. An R2 value of 0 indicates that the model does not explain any of the variances in the dependent variable. R2 values between 0 and 1 indicate the proportion of the variance that is explained by the model. R2 is calculated using the formula 4.2:

$$R2 = 1 - \frac{RSS}{TSS}, \tag{4.2}$$

where the sum of squares of residuals is $RSS = \Sigma(y - \hat{y})^2$, the total sum of squares is $TSS = \Sigma(y - \bar{y})^2$. y, $\hat{y}$, and $\bar{y}$ are the actual, predicted, and mean values of features respectively. Residuals refer to the disparity between the observed values and the predicted values.

4.3 Training

4.3.1 Settings

Each ML model has hyperparameters which by tweaking them requirements like memory, time, or accuracy are met. Hyperparameter tuning is an important phase of building a neural network. The most common parameters of the model are optimizers, the number of hidden layers, the number of nodes in each layer, loss function etc.

Moreover, besides the architecture of the model, also the shape of the input data enables *window size* to be used for tuning and discovering the optimal size of the window in the model.

Furthermore, grid search is utilized that seeks to identify the optimal hyperparameters for a model through the assessment of its performance on a grid of potential values. In grid search, a specific set of hyperparameters is established and the model is trained and assessed for each possible combination of the hyperparameter values on a grid. The performance of the model is assessed using a preferred evaluation metric and the combination of hyperparameters that yields the best performance is selected as the optimal set of hyperparameters.

Although grid search can be time-consuming due to the extensive number of hyperparameters and their potential values. In order to alleviate the load, as well as considering the redundancy of the models only the following few parameters are tested for each hyperparameter:

- For LSTM model; number of layers in LSTM layer 2, 3, number of neurons in each LSTM layer 64, 128, number of neurons in first fully connected layer 128, 256.

- For SIREN model; number of layers in SIREN layer 3, 4, number of neurons in each SIREN layer 64, 128, number of neurons in first fully connected layer 16, 32, 64.

4.3.2 Hardware Setup

Python programming language is used for analysis and model building. The machine used to train the neural network has 11th Gen Intel(R) Core(TM) i7-11800H @

2.30GHz as CPU and NVIDIA GeForce RTX 3070 with 8GB memory as GPU, 16GB RAM and is running on 64bits. Training is done in GPU.

4.4 Metrics

4.4.1 Loss functions

In machine learning, a loss function is a function used to assess the difference between the predicted output of a model and the actual output. The purpose of a loss function is to reduce this difference or "loss" as much as possible during model training. The choice of a loss function depends on the nature of the problem and the type of output being predicted. For instance, binary cross-entropy [76] is a commonly used loss function for binary classification problems, while mean squared error (MSE, eq. 4.3) is often used for regression problems. The loss function determines the effectiveness of a model, and minimizing the loss function is an important goal in machine learning model training.

The most famous loss functions for regression models are calculated as:

$$MSE = \frac{\Sigma(y - \hat{y})^2}{n} \tag{4.3}$$

$$MAE = \frac{\Sigma\,|y - \hat{y}|}{n} \tag{4.4}$$

$$MAPE = \frac{1}{n}\Sigma\frac{|y - \hat{y}|}{y}, \tag{4.5}$$

where y and $\hat{y}$ are the actual and predicted values respectively while n is the number of samples.

MSE (eq. 4.3) is the mean of the squared differences between the actual and predicted values. Large errors are penalized more heavily which makes the optimization process smoother. MAE (eq. 4.4) is the mean of the absolute differences between the actual and predicted values. All errors are treated as equally important. MAPE (eq. 4.5) calculates the absolute percentage difference between the actual and predicted values. The most important distinction of MAPE from others is that it can be used with any scale since it represents the values as percentages. The limitation of MAPE is that it is sensitive to extreme values and zero for having it in the denominator. All of these functions are evaluated in experimentation and while MAE and MSE are taken as loss functions for training.

Chapter 5

Results

In this chapter, results from different experiments are explained. The baseline is defined first which is accompanied by the results coming from various experiments done for robustness and sensibility checks. The striking differences and similarities between the experiment results are drawn to attention. Rich visuals and tables are provided for the readers.

5.1 Baseline

To test different scenarios, the baseline is defined and the models are trained and tested based on this baseline setting. In our case baseline is defined by considering experiments with particles that are sized small and large as training and experiments with medium size particles as the test dataset. Moreover, in the training phase, all of the experiments are utilized (100%). It is important to mention that both models with 5 different window sizes 10, 15, 20, 30, 50 and 2 different loss functions MAE, MSE are trained to define the baseline making up a total of 20 best-result for each metric MAE, MAPE, MSE, R2. Table 5.1 shows the best metrics obtained using grid search for each model, window size and loss function. Although the best values are near and quite low for both models SIREN performed slightly better by 3 out of 4 metrics. From the table 5.1, it seems that there is no strong relationship between window sizes and performance. Overall, for building the table 400 models were created. Each cell represents the optimal result from a specific configuration of hyperparameters given model, window size, loss function, and one of the 4 metrics. It can be noticed also that using MSE as the loss function gives a moderate advantage, which is more clear from training progress too. For the LSTM model, figures 5.1 and 5.2 shows the decrease in loss over the epoch in training and validation respectively. The same type of graphs is plotted for the SIREN model as well. From the graphs, it is evident that training takes fewer epochs and loss

dropped more rapidly by using MSE.

Table 5.1: Best metrics for baseline.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04697	0.16750	0.00388	0.90054
		MSE	0.04754	0.18098	0.00395	0.89879
	15	MAE	0.04338	0.17709	0.00319	0.91370
		MSE	0.04410	**0.16038**	0.00323	0.91244
	20	MAE	0.04692	0.18723	0.00361	0.89762
		MSE	0.04414	0.17402	0.00327	0.90706
	30	MAE	0.04071 +1.35%	0.18411	0.00272 +8.99%	0.91393 -1.5%
		MSE	0.04330	0.18967	0.00315	0.90172
	50	MAE	0.05180	0.22871	0.00408	0.83799
		MSE	0.04953	0.21180	0.00405	0.83876
SIREN	10	MAE	0.04957	0.26725	0.00371	0.90582
		MSE	0.04499	0.25282	0.00310	0.92077
	15	MAE	0.05088	0.28078	0.00377	0.90063
		MSE	0.04307	0.23581	0.00275	**0.92768**
	20	MAE	0.05376	0.24404	0.00426	0.88070
		MSE	0.04532	0.22036	0.00289	0.91988
	30	MAE	0.05389	0.22434	0.00428	0.86839
		MSE	**0.04016**	0.20413	**0.00249**	0.92267
	50	MAE	0.05115	0.16932 +5.58%	0.00416	0.83698
		MSE	0.05220	0.26536	0.00360	0.85889

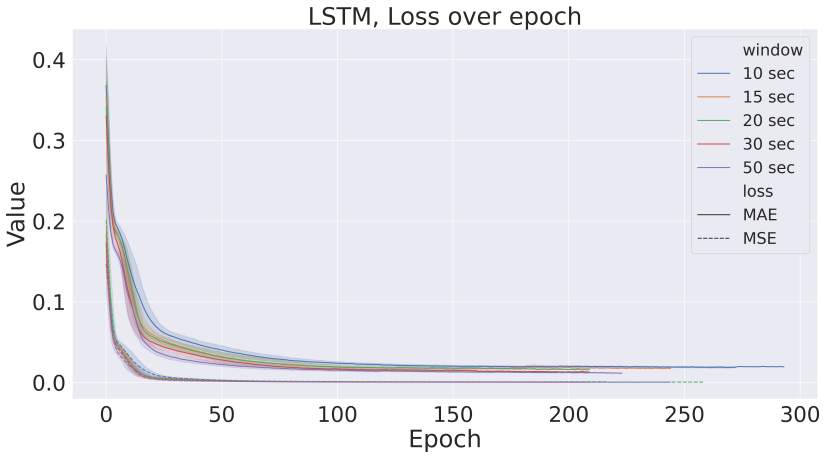

Figure 5.1: Training loss progress over epoch.

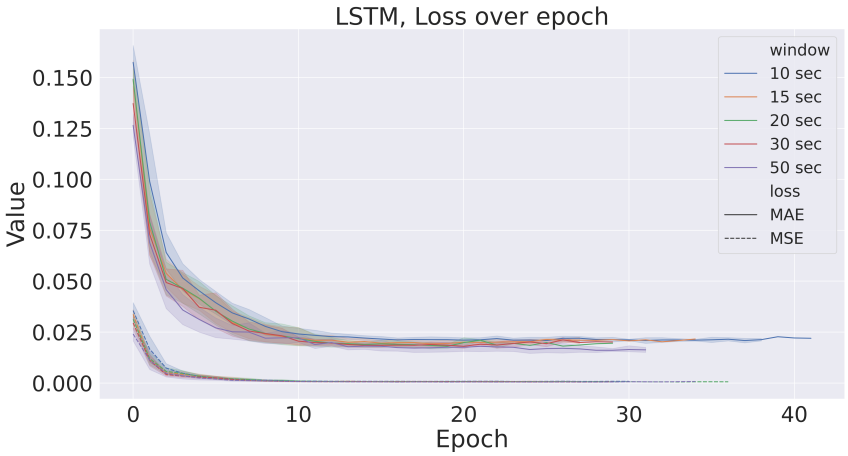

Figure 5.2: Validation loss progress over epoch.

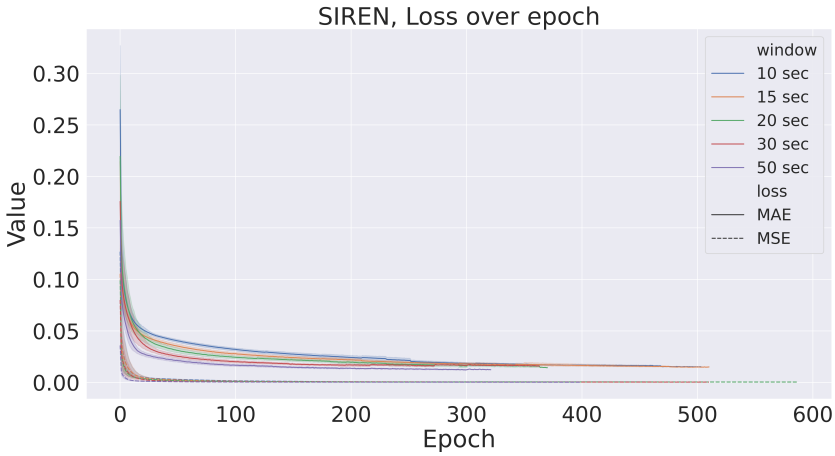

Figure 5.3: Training loss progress over epoch.

Figure 5.4: Validation loss progress over epoch.

To prove if the suggested methodology performs well in other settings model is trained and tested each time continuously with specific parameters and comparisons are made between the results and the baseline. On this matter first, sensitivity analysis is done by changing the experiments containing training and test datasets. From 3 particle sizes, it is possible to have 3 permutations of the train-test split, each time keeping one size only as the test and using the rest for training. This way one can measure the effect of particle size on the development of the model. A similar analysis is also performed for the ratio feature where samples with one type of ratio are completely removed from the training and are present in the testing dataset.

Yet another good method for sensitivity analysis is done by changing the number of experiments inside the training dataset, every time removing 25% of the experiments from the training dataset and then performing the training.

Progress of training and table of the best metrics present for each experiment accordingly. Test tables are constructed using the same strategy as was for the baseline.

5.2 Changing Compositions of Datasets based on Sample Size

The first sensibility analysis is done by tweaking the training dataset based on the sizes of the particles. In the baseline, the training dataset contains small and large particles. Thus, by changing the content of the training set two times we can have other 2 combinations of the train-test split. By doing this I measure the sensibility of my method against the different sample sizes. In other words, answering the

questions; Does my method performs well with different compositions? Is there any particle size that hugely affects the models' stability? From now on permutations of different test datasets are called permutations and baseline permutation is called permutation 1.

5.2.1 Permutation 2 (small particles for testing)

The first set of experiments is done by training the models with medium-sized and large-sized particles and testing it on the dataset with small particles only. Progress over the epoch does not show any abnormalities and loss quickly drops very close to 0 (5.5, 5.6, 5.7, 5.8).

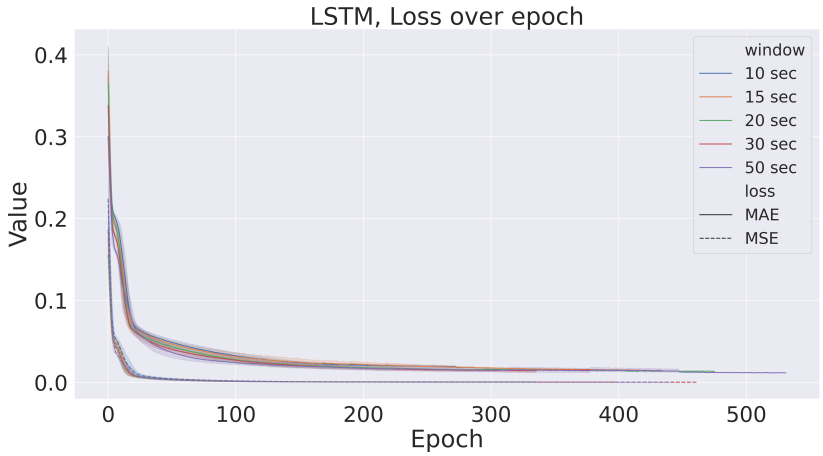

Figure 5.5: Training loss progress over epoch in permutation 2.

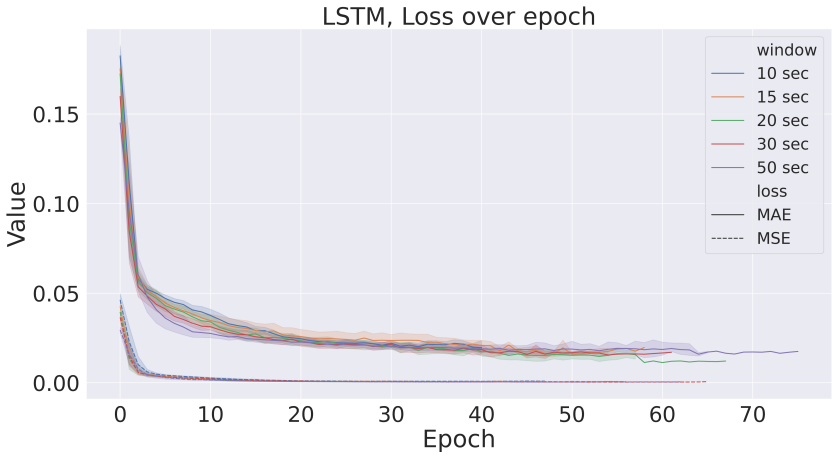

Figure 5.6: Validation loss progress over epoch permutation 2.

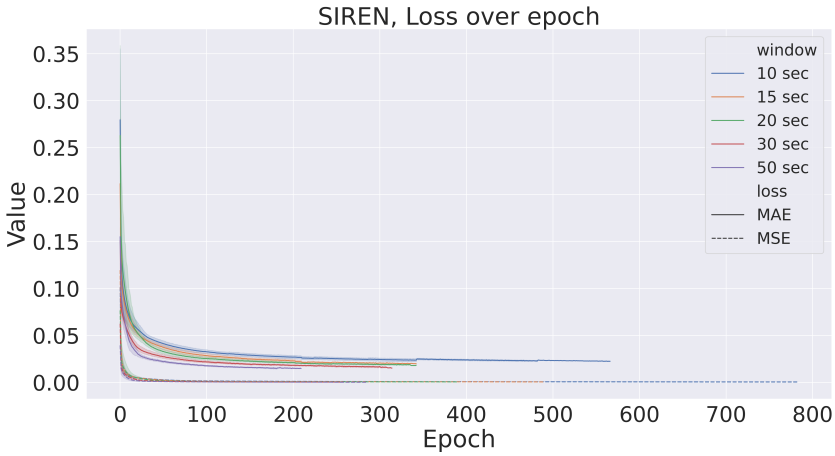

Figure 5.7: Training loss progress over epoch permutation 2.

Figure 5.8: Validation loss progress over epoch permutation 2.

The best results table (5.2) shows that LSTM performed much better in this particular case, winning four out of four metrics. Specifically, it is once again shown that in most cases having an MSE loss function makes models more robust. In general, both models have less than 0.2 MAPE and more than 0.9 R2 score which makes them good enough. After this, it becomes clear that not including the samples with small particles in training does not affect the results so much.

Table 5.2: Best metrics for the second permutation of the training dataset (medium and large particles for training).

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.05725	0.15301	0.00519	0.93938
		MSE	0.05862	0.17455	0.00504	0.94210
	15	MAE	0.05189	0.15727	0.00444	0.94685
		MSE	0.05651	0.17355	0.00484	0.94212
	20	MAE	0.05439	0.16193	0.00458	0.94212
		MSE	0.04762	0.15017	**0.00343**	**0.95720**
	30	MAE	0.06333	0.19227	0.00625	0.91384
		MSE	**0.04670**	**0.14548**	0.00346	0.95332
	50	MAE	0.08435	0.22342	0.01047	0.82820
		MSE	0.05704	0.18023	0.00477	0.92371
SIREN	10	MAE	0.08523	0.21961	0.01044	0.88152
		MSE	0.10660	0.33739	0.01407	0.83650
	15	MAE	0.08869	0.25877	0.01149	0.86391
		MSE	0.08485	0.25600	0.01052	0.87510
	20	MAE	0.06549	0.21189	0.00705	0.91324
		MSE	0.07109	0.19921	0.00711	0.91166
	30	MAE	0.05751	0.15079 +3.65%	0.00604	0.91882
		MSE	0.08785	0.25754	0.01051	0.85732
	50	MAE	0.06445	0.26501	0.00685	0.88973
		MSE	0.05465 +17.03%	0.15790	0.00489 +42.75%	0.92061 -3.97%

5.2.2 Permutation 3 (large particles for testing)

In one of the previous chapters while analyzing the dataset we noticed that samples with large-sized particles were acting in a slightly different way in the rig. In this particular permutation where the aforementioned samples are kept for the test dataset, this hypothesis is analyzed whether they are making a huge difference in the results of the model or not.

Upon first inspections of training processes (5.9, 5.10, 5.11, 5.12), it becomes evident that the necessary epochs are less than the previous permutations to finish the training, especially for LSTM model. This can be explained by the difference between the similarity between small and medium and the similarity between large and other particles. Having more similar datasets makes the training loss reach low stable values faster, whereas even a slight difference prolongs the time because now the model has seen more diverse data and needs to adapt to that too.

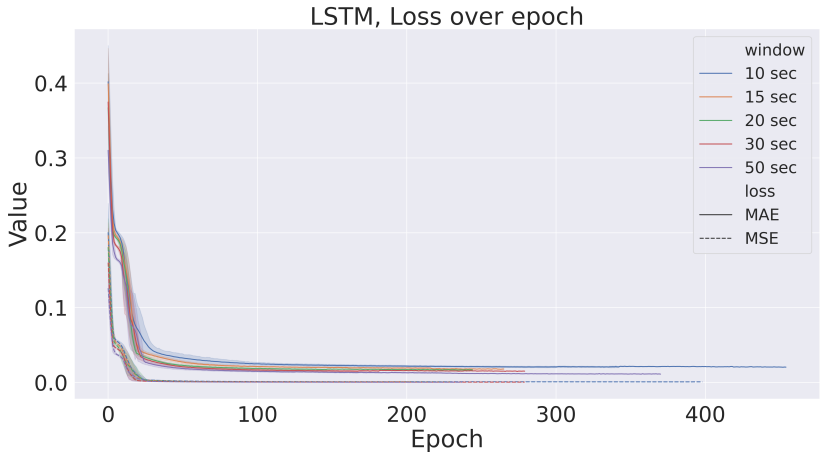

Figure 5.9: Training loss progress over epoch in permutation 3.

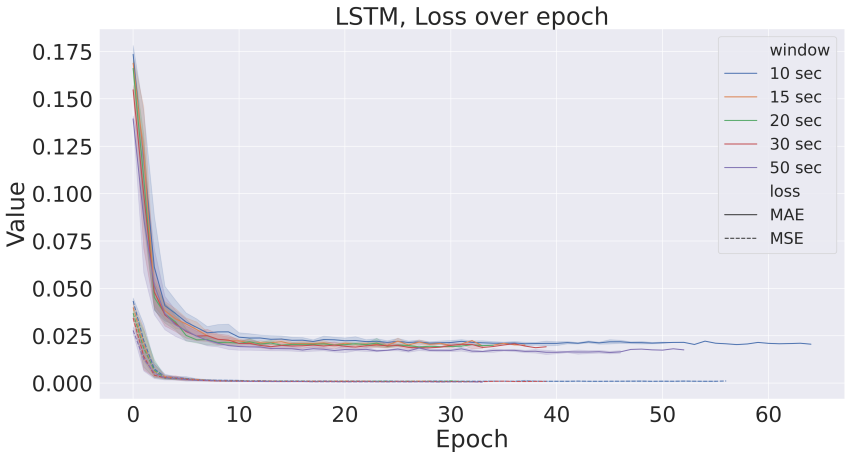

Figure 5.10: Validation loss progress over epoch permutation 3.

Figure 5.11: Training loss progress over epoch permutation 3.

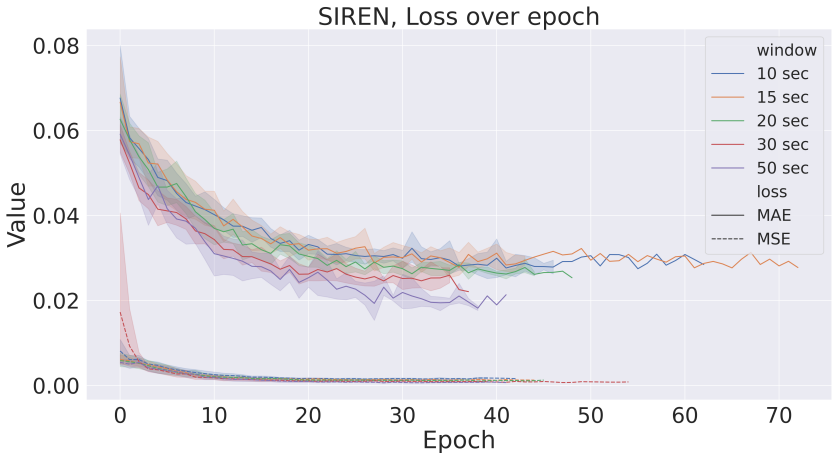

Figure 5.12: Validation loss progress over epoch permutation 3.

From the test results (table 5.3), we can prove true our hypothesis of "having the different particle-sized dataset for test affects the model fitness". Moreover, unexpected heavy underfitting can be seen. It can be immediately noticed that MAPE and R2 scores performed quite badly for LSTM model, having more than 0.4 and less than 0.36 respectively. In the field, these results are unacceptable for considering a model a fit. Results for SIREN are even much worse: having a lot of negative R2 and near 1 MAPE scores. These results also prove that relying only on one metric can be deceiving to see the fitness of the model.

Table 5.3: Best metrics for the third permutation of the training dataset (small and medium particles for training).

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	**0.12302**	**0.40404**	**0.02176**	**0.35864**
		MSE	0.13638	0.42992	0.02661	0.23393
	15	MAE	0.13271	0.45339	0.02758	0.13056
		MSE	0.13725	0.44996	0.02692	0.14811
	20	MAE	0.13658	0.48341	0.02636	0.11080
		MSE	0.13404	0.46874	0.02676	0.11053
	30	MAE	0.12998	0.48078	0.02418	0.08226
		MSE	0.12980	0.45816	0.02536	0.03100
	50	MAE	0.13516	0.50895	0.02538	-0.25955
		MSE	0.12864	0.48256	0.02388	-0.17783
SIREN	10	MAE	0.15667	0.57089	0.03349	0.01810
		MSE	0.16229	0.68034	0.03281	0.03286
	15	MAE	0.17682	0.93135	0.03762	-0.17379
		MSE	0.14838+20.62%	0.56270+39.27%	0.03107 +42.78%	0.03416-949.95%
	20	MAE	0.17969	0.90445	0.04055	-0.33529
		MSE	0.16465	0.68289	0.03601	-0.19189
	30	MAE	0.20600	1.24866	0.04707	-0.78287
		MSE	0.16098	0.71439	0.03453	-0.32448
	50	MAE	0.16983	0.76668	0.03663	-0.81685
		MSE	0.17484	0.79778	0.03753	-0.89143

5.3 Reducing Size of Training Dataset

To analyze the importance of dataset size used in training to achieve better performance number of training samples decreased 3 times, every time randomly eliminating 25% more than the previous time, i.e, 25%, 50%, and finally 75%. The remaining samples contain accordingly 75%, 50%, and 25% of the training dataset of baseline. After the analysis, it becomes obvious whether similar fitness can be achieved by using a smaller dataset. It is particularly useful to test the robustness of the methodology, proving that the model can perform well even in a lack of data.

75%

From the graphs (5.13, 5.14, 5.15, 5.16) of the progress of training, we can see that there is no evident change for LSTM model, while for SIREN number of epochs to complete the training is slightly increased. It is also noticed that using MAE as a loss function of SIREN results in unstable optimization and higher validation loss at the end (figure 5.16).

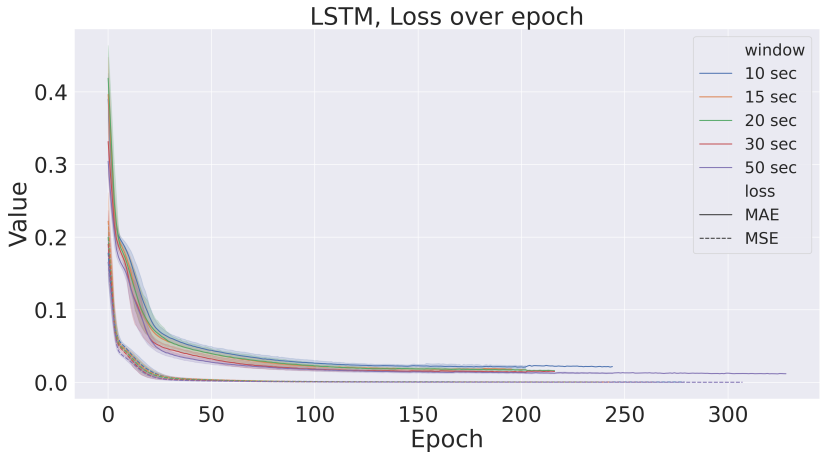

Figure 5.13: Training loss progress over epoch by using 75% of data.

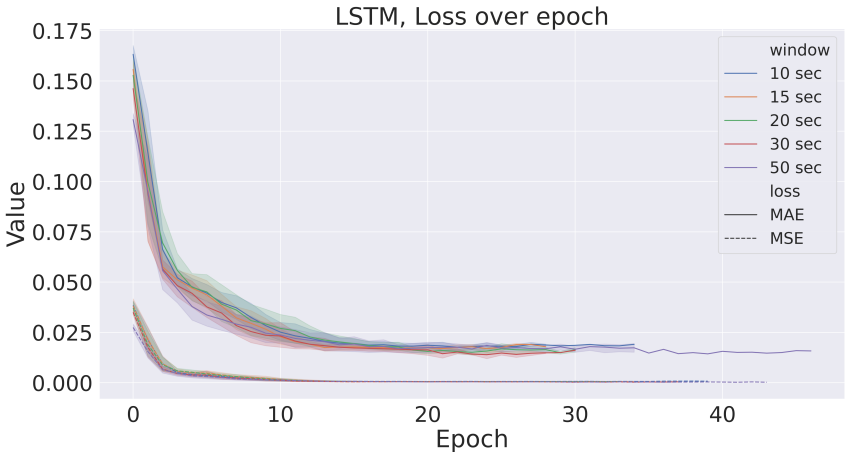

Figure 5.14: Validation loss progress over epoch by using 75% of data.

Figure 5.15: Training loss progress over epoch by using 75% of data.

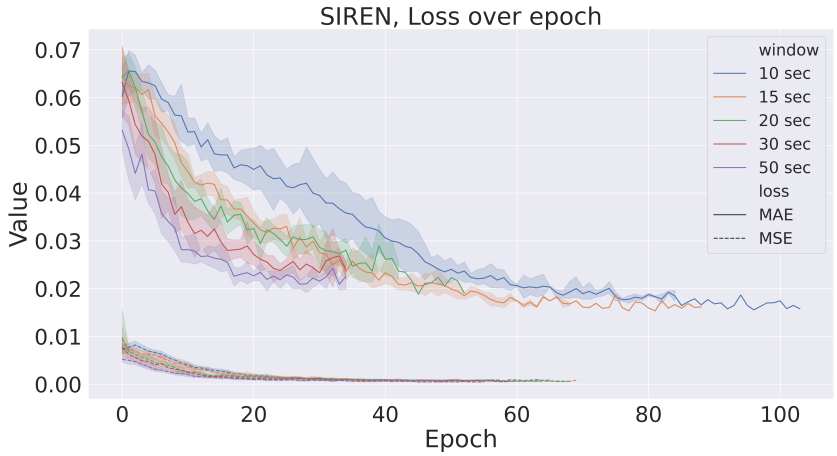

Figure 5.16: Validation loss progress over epoch by using 75% of data.

Table 5.4 shows the performance of the model stays stable (very good) with 25% fewer samples in training. There is no noticeable difference between the metrics of the two models, where SIREN negligibly performed better specifically for the mid-window sizes (15, 20, 30).

Table 5.4: Best metrics after using 75% of original data in training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04366	0.16177	0.00309	0.92270
		MSE	0.04430	0.16973	0.00354	0.90966
	15	MAE	0.04052+1.19%	0.15920 +2.13%	0.00271+8.6%	0.92667-0.83%
		MSE	0.04633	0.17874	0.00352	0.90434
	20	MAE	0.04218	0.17374	0.00297	0.91448
		MSE	0.04635	0.18483	0.00366	0.89425
	30	MAE	0.04596	0.18794	0.00338	0.89226
		MSE	0.04451	0.20135	0.00329	0.89560
	50	MAE	0.04206	0.19491	0.00289	0.88556
		MSE	0.05233	0.22614	0.00429	0.83010
SIREN	10	MAE	0.05510	0.15886	0.00438	0.88749
		MSE	0.05183	0.22666	0.00387	0.90278
	15	MAE	0.05013	0.22975	0.00349	0.90704
		MSE	0.04009	0.21262	**0.00249**	**0.93434**
	20	MAE	0.05340	0.20546	0.00376	0.89525
		MSE	0.04190	0.23073	0.00261	0.92778
	30	MAE	**0.04005**	**0.15588**	0.00276	0.91558
		MSE	0.04834	0.23117	0.00332	0.89769
	50	MAE	0.05637	0.23370	0.00434	0.82944
		MSE	0.06184	0.29354	0.00464	0.81704

50%

First glance at the training progresses (5.17, 5.18, 5.19, 5.20) reveals that the decreasing the training size results in prolonged processes. Having fewer data to reach the low and stable validation loss lasts longer. Similar to the previous case, using MAE as a loss function results in less stable optimization.

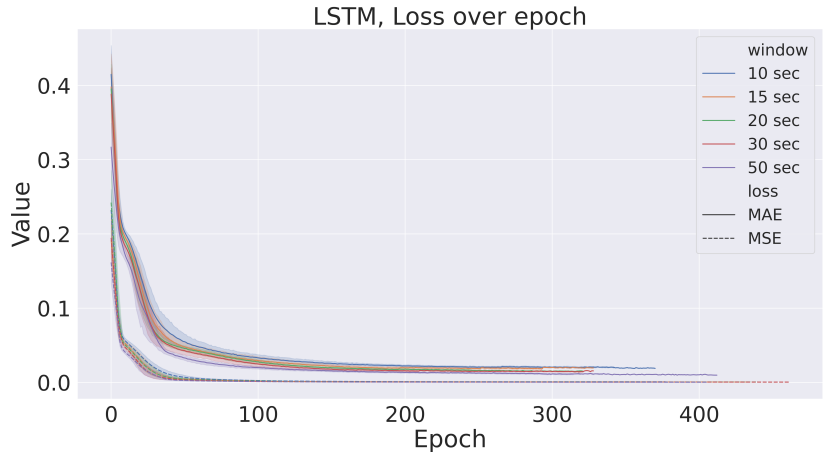

Figure 5.17: Training loss progress over epoch by using 50% of data.

Figure 5.18: Validation loss progress over epoch by using 50% of data.

Figure 5.19: Training loss progress over epoch by using 50% of data.

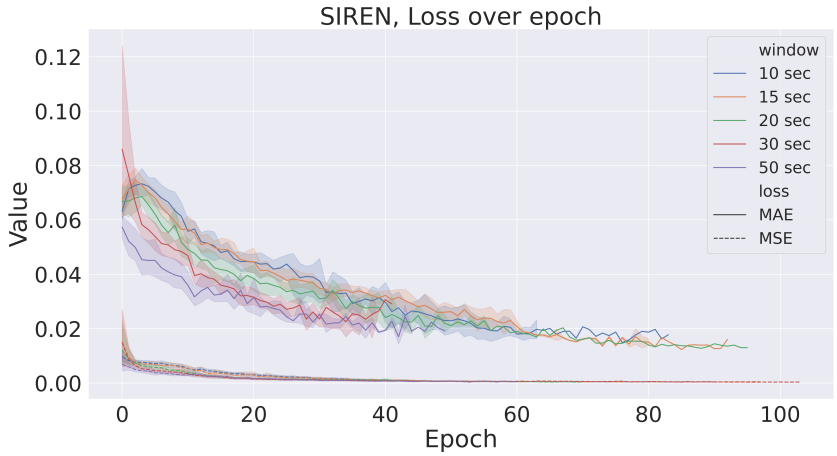

Figure 5.20: Validation loss progress over epoch by using 50% of data.

Results presented in table 5.5 show even better results in each metric, reaching as high as 0.954 R2 score and dropping below 0.14 MAPE score. The outcome of the test in this phase suggests that it is enough to use only half of the training data to reach optimal results.

Table 5.5: Best metrics after using 50% of original data in training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.03919	**0.13927**	0.00286	0.92731
		MSE	0.03717+8.97%	0.15578	0.00231	0.94089-1.36%
	15	MAE	0.04468	0.17326	0.00330	0.91103
		MSE	0.03743	0.15502	0.00231+31.07%	0.93873
	20	MAE	0.04120	0.17110	0.00261	0.92748
		MSE	0.04161	0.17414	0.00287	0.91941
	30	MAE	0.04212	0.18547	0.00288	0.90847
		MSE	0.04646	0.19240	0.00336	0.89454
	50	MAE	0.04819	0.20138	0.00351	0.86229
		MSE	0.04538	0.21238	0.00315	0.87402
SIREN	10	MAE	0.05337	0.21222	0.00381	0.90509
		MSE	0.04232	0.16925	0.00264	0.93470
	15	MAE	0.04458	0.21892	0.00284	0.92674
		MSE	**0.03411**	0.16091	**0.00176**	**0.95369**
	20	MAE	0.05772	0.19623	0.00430	0.88125
		MSE	0.04137	0.18083	0.00269	0.92714
	30	MAE	0.06170	0.19196	0.00601	0.81315
		MSE	0.04585	0.22145	0.00322	0.90205
	50	MAE	0.04967	0.18354	0.00419	0.83723
		MSE	0.04635	0.15085 +8.32%	0.00374	0.85066

25%

Graphs below (5.21, 5.22, 5.23, 5.24) proves that time to complete the training increases when the size of the training data decreases. However, the same pattern can be seen in training progress where, particularly in the SIREN model, MSE leads to better training.

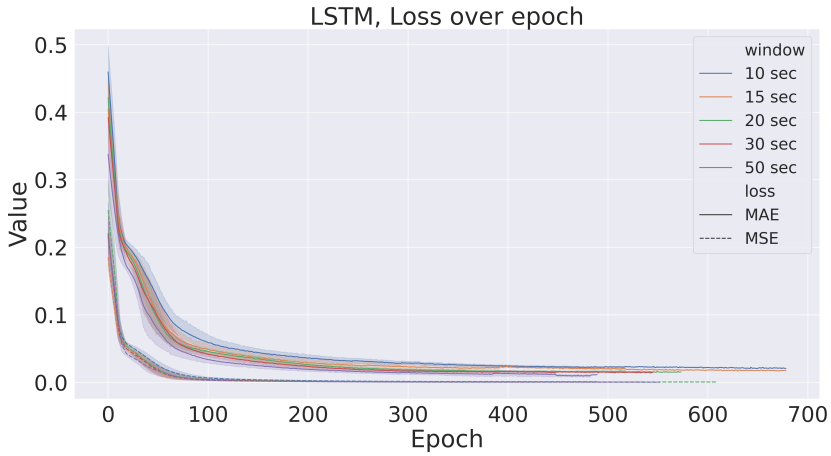

Figure 5.21: Training loss progress over epoch by using 25% of data.

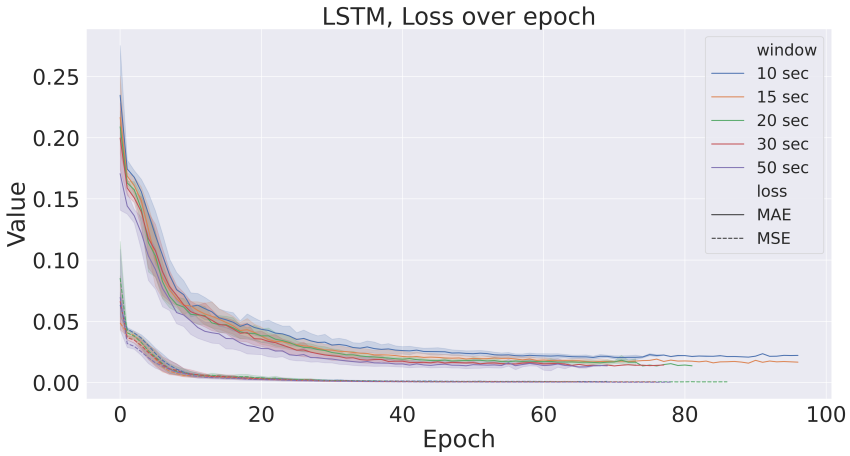

Figure 5.22: Validation loss progress over epoch by using 25% of data.

Figure 5.23: Training loss progress over epoch by using 25% of data.

Figure 5.24: Validation loss progress over epoch by using 25% of data.

Even though metrics show a little drawback in the fitness of the model, the test results are still very good, over 0.9 R2 less than 0.15 MAPE scores along with close MAE and MSE (table 5.6). It is good to mention that in the case of using 25% of the training data, LSTM performs slightly better than the SIREN model.

Table 5.6: Best metrics after using 25% of original data in training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04341	0.16859	0.00322	0.91840
		MSE	0.04347	0.16446	0.00324	0.91752
	15	MAE	**0.04315**	0.18178	**0.00302**	**0.91897**
		MSE	0.04797	0.18938	0.00366	0.90069
	20	MAE	0.04364	0.15949+10.97%	0.00323	0.90921
		MSE	0.04452	0.17536	0.00327	0.90747
	30	MAE	0.04499	0.18559	0.00317	0.90146
		MSE	0.04688	0.18682	0.00362	0.88786
	50	MAE	0.04879	0.20811	0.00360	0.86049
		MSE	0.05081	0.19810	0.00392	0.84744
SIREN	10	MAE	0.07159	0.21874	0.00734	0.81629
		MSE	0.05822	0.23120	0.00502	0.87792
	15	MAE	0.06306	0.19462	0.00632	0.83585
		MSE	0.05834	0.20849	0.00487	0.87192
	20	MAE	0.04992	0.17458	0.00401	0.89061 -3.19%
		MSE	0.06005	0.21264	0.00503	0.85983
	30	MAE	0.06307	0.22171	0.00550	0.83194
		MSE	0.05621	0.23602	0.00432	0.86859
	50	MAE	0.06111	0.19073	0.00586	0.77424
		MSE	0.04807 +11.4%	**0.14373**	0.00336+10.97%	0.87027

5.4 Reducing the Size of Datasets based on Particle Ratio

To analyze the sensitivity of the model towards the particle ratios, they are removed from the training dataset 4 times, each time dropping all the samples containing one of the *{0.4, 0.425, 0.45, 0.475}* ratios. Best metrics are then calculated for each set of training processes using the same testing methodology used before and tables are drawn for the comparison.

5.4.1 Training without particle ratio of 0.4

Training progress after dropping the samples with a particle ratio of 0.4 does not show striking differences (5.25, 5.26, 5.27, 5.28). Quite the contrary it shows similarity in terms of reaching the minima, both for the training and validation steps. Meanwhile, the results coming from the test table 5.7 suggest that removing the specified particles from the training set does not make the model worse. The SIREN model outperformed the counterpart by having greater R2 and lower MSE and MAE. However, its lowest MAPE score is over 22% which is an indication of poor fitness.

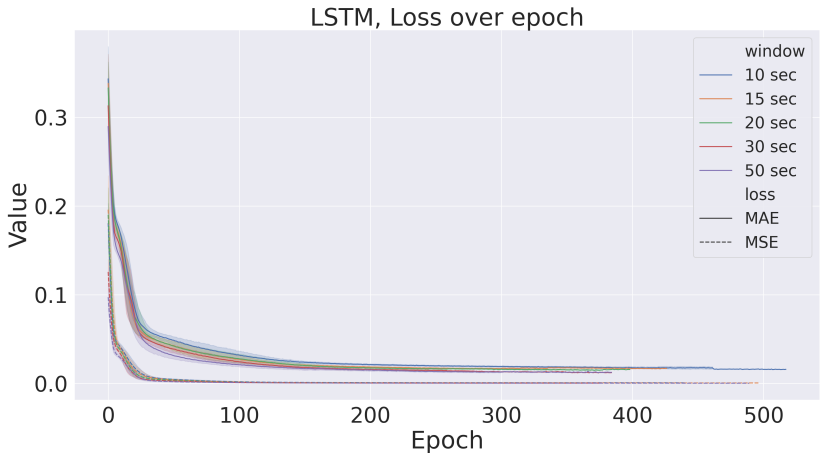

Figure 5.25: Training loss progress over epoch with the absence of 0.4 particle ratio.

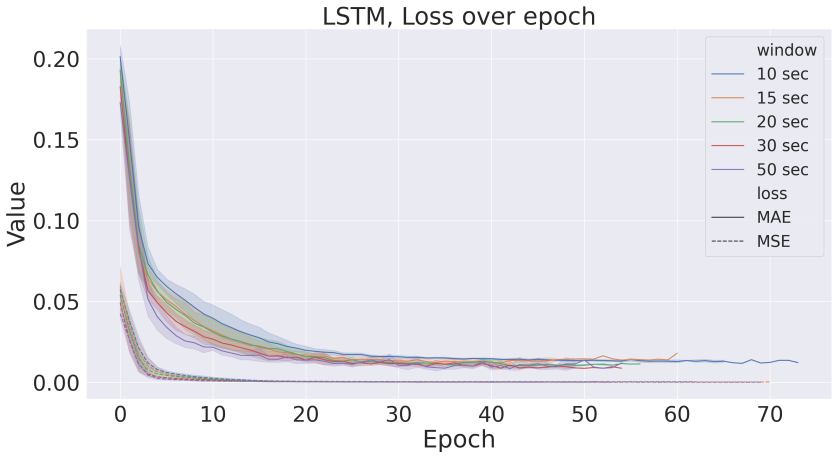

Figure 5.26: Validation loss progress over epoch with the absence of 0.4 particle ratio.

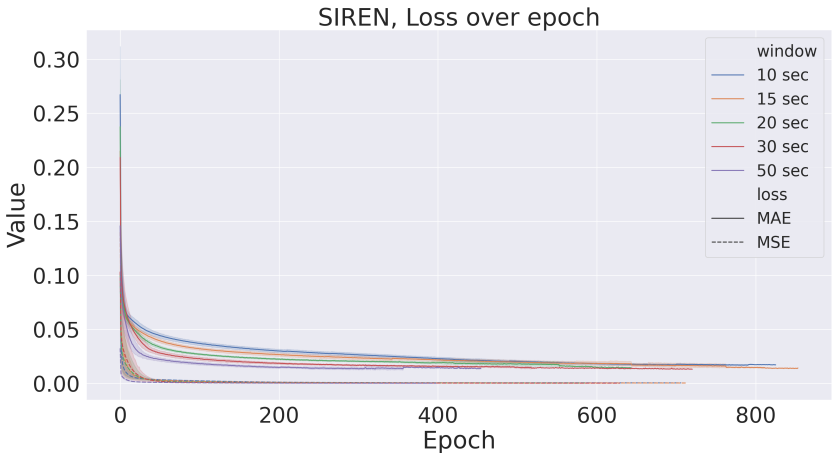

Figure 5.27: Training loss progress over epoch with the absence of 0.4 particle ratio.

Figure 5.28: Validation loss progress over epoch with the absence of 0.4 particle ratio.

Table 5.7: Best metrics after removing samples with particle ratio of 0.4 of original data in the training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04936	**0.17637**	0.00389	0.90666 -4.05%
		MSE	0.05058	0.18820	0.00430	0.89894
	15	MAE	0.05008	0.17817	0.00430	0.89425
		MSE	0.05349	0.19829	0.00456	0.88503
	20	MAE	0.04805	0.18684	0.00360+47.75%	0.90549
		MSE	0.04800+24.02%	0.17954	0.00379	0.89893
	30	MAE	0.05316	0.21425	0.00428	0.87589
		MSE	0.04875	0.19992	0.00366	0.89372
	50	MAE	0.05499	0.21901	0.00471	0.82501
		MSE	0.05388	0.21772	0.00437	0.84093
SIREN	10	MAE	**0.03870**	0.25762	**0.00244**	**0.94340**
		MSE	0.05493	0.35410	0.00471	0.89051
	15	MAE	0.04662	0.30073	0.00337	0.91900
		MSE	0.04224	0.24521	0.00269	0.93419
	20	MAE	0.04777	0.23781	0.00354	0.90879
		MSE	0.04296	0.22557+27.9%	0.00282	0.92818
	30	MAE	0.05964	0.32032	0.00459	0.86671
		MSE	0.05772	0.29051	0.00451	0.87042
	50	MAE	0.06300	0.35560	0.00467	0.82993
		MSE	0.06083	0.30060	0.00481	0.82368

5.4.2 Training without particle ratio of 0.425

Regarding the training process, this version of the training takes fewer epochs than the previous case. However, in terms of reaching the minima they both acted similarly (5.29, 5.30, 5.31, 5.32). In terms of test results presented in table 5.8, we can see that it is a slightly better version of 0.4 in terms of error metrics, where R2 is quite close.

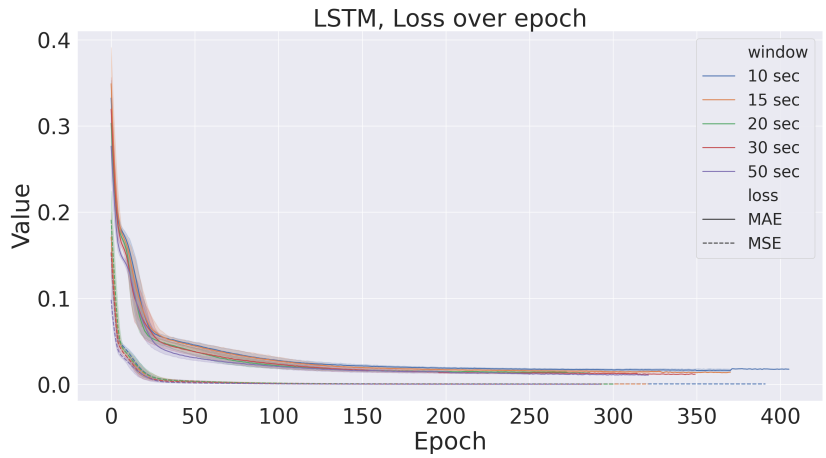

Figure 5.29: Training loss progress over epoch with the absence of 0.425 particle ratio.

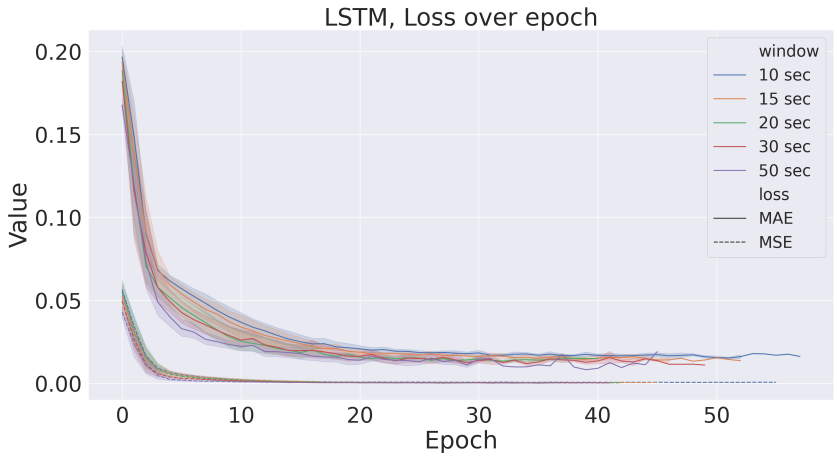

Figure 5.30: Validation loss progress over epoch with the absence of 0.425 particle ratio.

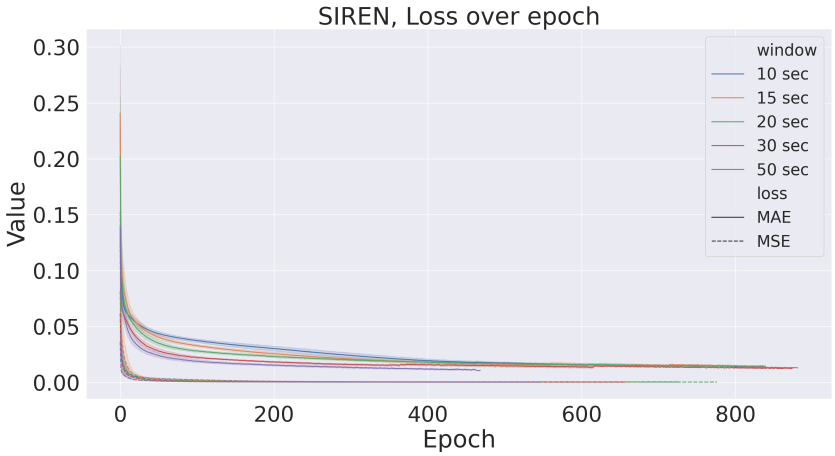

Figure 5.31: Training loss progress over epoch with the absence of 0.425 particle ratio.

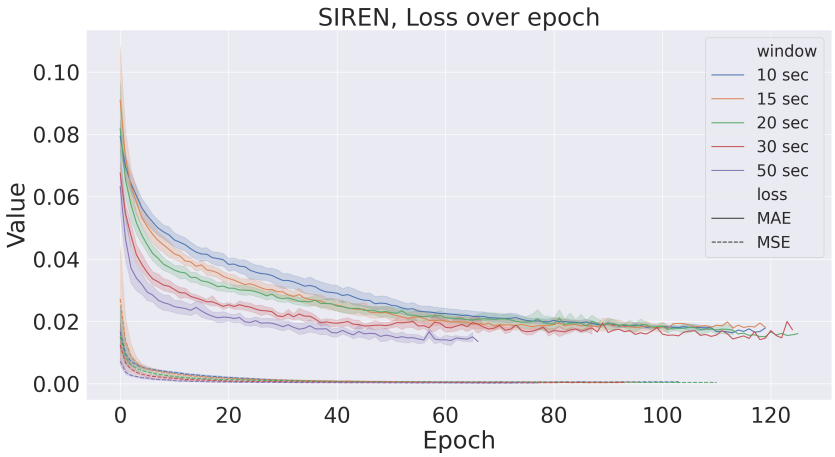

Figure 5.32: Validation loss progress over epoch with the absence of 0.425 particle ratio.

Table 5.8: Best metrics after removing samples with particle ratio of 0.425 of original data in the training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04357	**0.15760**	0.00300+67.66%	0.92271 -2.27%
		MSE	0.04725	0.18494	0.00358	0.90593
	15	MAE	0.04512	0.18406	0.00337	0.90900
		MSE	0.04985	0.19287	0.00405	0.88982
	20	MAE	0.04833	0.18591	0.00374	0.89133
		MSE	0.04624	0.18609	0.00351	0.89938
	30	MAE	0.04287+19.42%	0.18368	0.00302	0.90428
		MSE	0.05236	0.20126	0.00401	0.87318
	50	MAE	0.05738	0.24692	0.00502	0.79943
		MSE	0.05036	0.22617	0.00381	0.84711
SIREN	10	MAE	0.04600	0.26057	0.00331	0.91680
		MSE	0.05453	0.32464	0.00443	0.88499
	15	MAE	0.04032	0.19365+22.88%	0.00244	0.93555
		MSE	0.03745	0.24204	0.00211	**0.94365**
	20	MAE	0.05066	0.30210	0.00375	0.89664
		MSE	0.03947	0.24514	0.00229	0.93702
	30	MAE	0.04677	0.21073	0.00311	0.90342
		MSE	0.05040	0.27721	0.00347	0.89283
	50	MAE	**0.03590**	0.21317	**0.00179**	0.92944
		MSE	0.04219	0.22798	0.00256	0.90012

5.4.3 Training without particle ratio of 0.45

For this particular case too striking differences are not visible in the training process (5.33, 5.34, 5.35, 5.36). Although we can see a marginal drop in R2 score in test results, other metrics are very close to the previous cases (5.9). This variation of the training process as well as SIREN with a window size of 30 seconds and MSE as a loss function outperformed LSTM slightly.

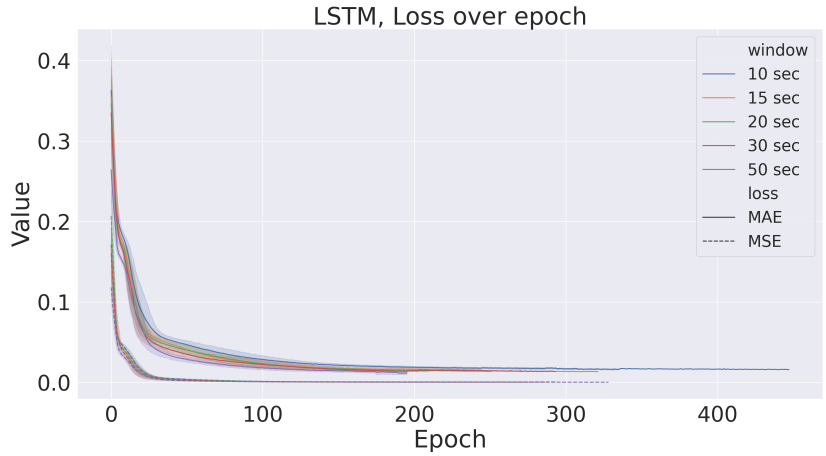

Figure 5.33: Training loss progress over epoch with the absence of 0.45 particle ratio.

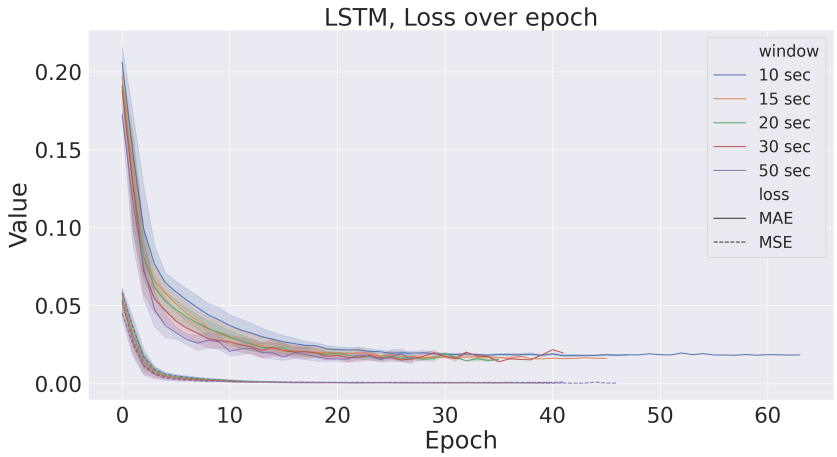

Figure 5.34: Validation loss progress over epoch with the absence of 0.45 particle ratio.

Figure 5.35: Training loss progress over epoch with the absence of 0.45 particle ratio.

Figure 5.36: Validation loss progress over epoch with the absence of 0.45 particle ratio.

Table 5.9: Best metrics after removing samples with particle ratio of 0.45 of original data in the training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04714	0.17092	0.00374	0.90391
		MSE	0.04692	0.16806	0.00359	0.90766
	15	MAE	0.04475	0.18772	0.00352	0.90596
		MSE	0.04936	0.19429	0.00412	0.88826
	20	MAE	0.04681	0.19349	0.00360	0.89777
		MSE	0.04284+14.38%	**0.16152**	0.00303+29.78%	0.91322 -1.66%
	30	MAE	0.04597	0.20028	0.00353	0.88722
		MSE	0.04957	0.20217	0.00405	0.87255
	50	MAE	0.05080	0.22188	0.00392	0.84369
		MSE	0.05154	0.21920	0.00402	0.83991
SIREN	10	MAE	0.05447	0.29046	0.00434	0.89222
		MSE	0.05326	0.37416	0.00413	0.89600
	15	MAE	0.04955	0.31507	0.00369	0.90340
		MSE	0.04534	0.30577	0.00311	0.91891
	20	MAE	0.04625	0.23656	0.00340	0.90501
		MSE	0.05324	0.30526	0.00385	0.89197
	30	MAE	0.06693	0.33043	0.00566	0.82409
		MSE	**0.03745**	0.19687+21.89%	**0.00234**	**0.92837**
	50	MAE	0.04214	0.21891	0.00237	0.90668
		MSE	0.04689	0.26119	0.00318	0.87614

5.4.4 Training without particle ratio of 0.475

Finally, the last ratio available in the training dataset is removed and models are trained using the rest of the samples. It can be observed from the progress that it took for these experiments, especially for the LSTM model to finish the training epochs (5.37, 5.38, 5.39, 5.40). There is no big variance from previous results in test results in table 5.10 while having 15 seconds window showed the top performance for both NN architectures.

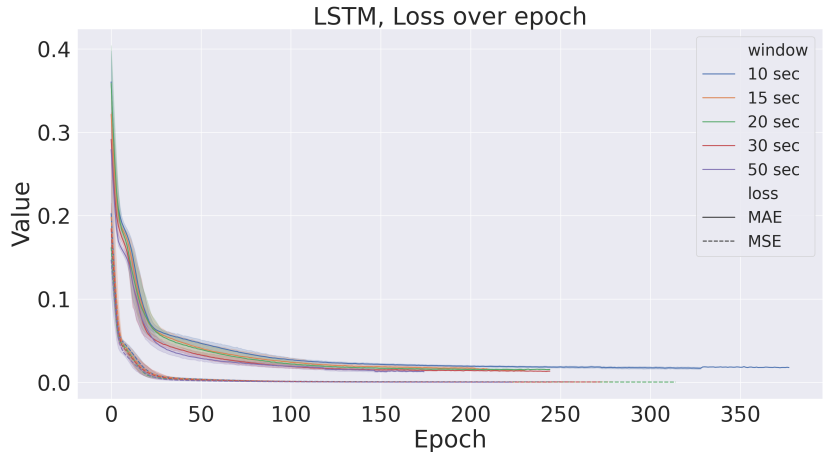

Figure 5.37: Training loss progress over epoch with the absence of 0.475 particle ratio.

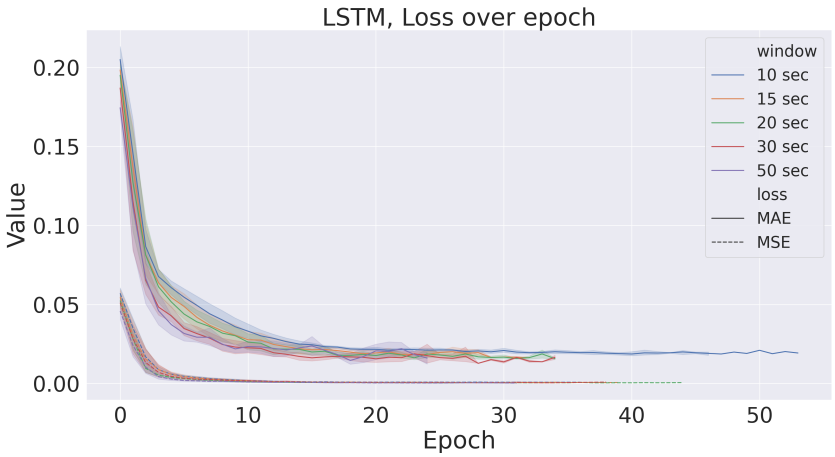

Figure 5.38: Validation loss progress over epoch with the absence of 0.475 particle ratio.

Figure 5.39: Training loss progress over epoch with the absence of 0.475 particle ratio.

Figure 5.40: Validation loss progress over epoch with the absence of 0.475 particle ratio.

Table 5.10: Best metrics after removing samples with particle ratio of 0.475 of original data in the training dataset.

MODEL	WINDOW	LOSS	METRIC			
			MAE	MAPE	MSE	R2
LSTM	10	MAE	0.04443	0.17390	0.00331	0.91413
		MSE	0.04678	0.18445	0.00352	0.90877
	15	MAE	0.04655	0.18096	0.00372	0.89899
		MSE	0.04203	**0.16752**	0.00287	0.92272-1.3%
	20	MAE	0.04090+3.85%	0.17643	0.00274+10.06%	0.92083
		MSE	0.04340	0.17970	0.00320	0.90952
	30	MAE	0.04345	0.18885	0.00331	0.89539
		MSE	0.04240	0.17950	0.00307	0.90264
	50	MAE	0.05806	0.24108	0.00498	0.80223
		MSE	0.04766	0.21272	0.00344	0.86325
SIREN	10	MAE	0.04406	0.31780	0.00326	0.91761
		MSE	0.04891	0.29287	0.00370	0.90639
	15	MAE	0.04289	0.25666	0.00269	0.92935
		MSE	**0.03938**	0.20344	**0.00249**	**0.93474**
	20	MAE	0.04540	0.27486	0.00309	0.91497
		MSE	0.04129	0.23606	0.00255	0.92947
	30	MAE	0.05910	0.28322	0.00474	0.85418
		MSE	0.04985	0.27116	0.00337	0.89535
	50	MAE	0.05173	0.20100 +19.99%	0.00389	0.84671
		MSE	0.07441	0.40886	0.00667	0.73964

5.5 Summary

Best models have the following hyperparameters (table 5.11) using the R2 as a main metric. It is evident from the table that every experiment has a very good performance except permutation 3 where samples with large-sized particles are left for testing. Both architectures have an equal share, meaning 6 SIREN and 6 LSTM acted as the best models. In terms of window sizes, 15-seconds is the most frequent while 50-seconds is not represented at all. The table proves once more that using MSE as a loss function is the most well-performed way. Another very evident finding is that LSTM models has always the same internal hyperparameters for the number of nodes in the first layers, the number of layers, and the number of nodes in the first fully connected layer, 64-3-256 respectively. Meanwhile, SIREN has 64-4-64 for the same parameters mentioned.

Figure 5.41 shows an example where the test sample (sample #1) is fed into the model. The model chosen is the best configuration from the table 5.11, an LSTM model with a window size of 20, loss function as MSE where the R2 score is 0.9572, the highest of all.

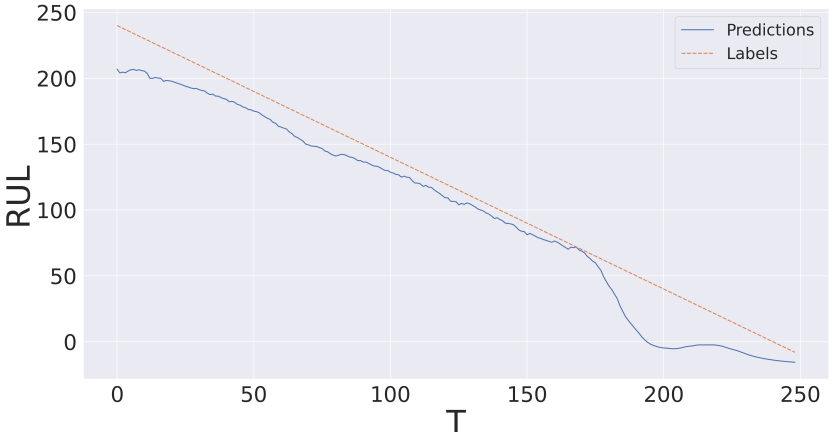

Figure 5.41: Sample prediction using the best model.

Table 5.11: Best results and their configuration based on R2. NiFL = Nodes in First Layers, L = Layers, NiFFCL = Nodes in First Fully Connected Layer.

				Internal hyperparameters			
Experiment	R2	Model	Window	Loss	NiFL	L	NiFFCL
Baseline	0.92768	SIREN	15	MSE	64	4	64
Perm 2	0.9572	LSTM	20	MSE	64	3	256
Perm 3	0.35864	LSTM	10	MAE	64	3	256
75%	0.93434	SIREN	15	MSE	64	4	64
50%	0.95369	SIREN	15	MSE	64	4	64
25%	0.91897	LSTM	15	MAE	64	3	256
Drop 0.4	0.9434	LSTM	10	MAE	64	3	256
Drop 0.425	0.94365	SIREN	15	MSE	64	4	64
Drop 0.45	0.92837	SIREN	30	MSE	64	4	64
Drop 0.475	0.93474	SIREN	15	MSE	64	4	64

Chapter 6

Conclusion and Future Improvements

Predictive maintenance has gained a lot of attention from the industry due to the capability of saving a lot of time, and thus money for businesses. Predicting the remaining useful life of a component might be the main point to avoid unintentional interrupts in the industrial process due to low tolerance for unnecessary breaks. The recent developments in the field of computer science and, specifically, exponential developments in Artificial Intelligence, make it easier to build efficient models that are capable of forecasting the remaining useful life of an asset precisely.

In this research thesis, I provide an elaborate background and state-of-the-art related to the predictive maintenance and usage of AI in the field. Publicly available case study data is used to test the arguments of the thesis. State-of-the-art artificial neural network architectures are tested and results are provided.

Overall, the results are promising for the models used. Especially, the recently developed architecture called SIREN is tested for the usage of predicting RUL in the filters and showed good performance. Different setups of training datasets are tested to see the robustness concerning the specific configuration of parameters of the dataset, such as particle sizes. The outcome of the experiments shows that the absence of samples with one type of particle size (large) in the training process affects the model, which in turn results in a very high overfit. By this a hypothesis asked during the analysis of the data is addressed.

Moreover, the response to whether changing the size of the training dataset affects or not, and how is provided by the results of the models after the dropping of random samples from the training dataset. With this regard, the overall fitness of the model is not changed which means that using as few as 25% of the original training samples does not take a toll on the model at all. Apart from random elimination, samples with specific ratios are dropped every time to create four

models and tested on the test dataset. Results show that the performance of the methodology stayed high no matter which particle ratio is absent from the training. This has a particular meaning that the models are not sensitive towards the particle ratio, where it can still be robust if there is any missing from training.

The results of the thesis prove that the specific AI models can be utilized in the scenario of predicting the RUL of a filter. However the dataset is coming from a physically built rig and the simulation is more likely to be very similar to the real examples, one cannot be sure how much it can be useful for real plants and filters. Future improvements can be done on this matter by testing the methodology on data collected by real processes and application of it to real-world examples.

Moreover, in terms of models' performance, however how good the performance of the models tested, different parameters can be tried, provided enough time and computational power. Trying on more combinations of dataset configurations can be a good start.

Bibliography

[1] Eker O.F. «A Hybrid Prognostic Methodology and its Application to Well-Controlled Engineering Systems (PhD thesis)». Cranfield University, 2015 (cit. on pp. i, 10, 11, 13, 19).

[2] Colin Lewis. *Industrial and business forecasting methods*. en. Oxford, England: Butterworth-Heinemann, Jan. 1982 (cit. on p. i).

[3] MIT Sloan Review. «GE's Big Bet on Data and Analytics». In: *MIT Sloan Review* (2016). URL: https://sloanreview.mit.edu/case-study/ge-big-bet-on-data-and-analytics/ (cit. on p. 2).

[4] TechTarget. *Predictive maintenance software points to machinery problems* (cit. on p. 2).

[5] European Commission, Directorate-General for Research, and Innovation. *Investment in research and innovation underpins the industrial transition of tomorrow*. Publications Office, 2021. DOI: doi/10.2777/297396 (cit. on p. 2).

[6] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. arXiv: 2204.06125 [cs.CV] (cit. on pp. 2, 3).

[7] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL] (cit. on p. 2).

[8] HENNING KAGERMANN, WOLF-DIETER LUKAS, and WOLFGANG WAHLSTER. *Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. Industriellen Revolution. VDI Nachrichten*. URL: %7Bhttps://www.dfki.de/fileadmin/user_upload/DFKI/Medien/News_Media/Presse/Presse-Highlights/vdinach2011a13-ind4.0-Internet-Dinge.pdf%7D (cit. on p. 5).

[9] URL: https://www.industry4cloud.net/industry40 (cit. on p. 6).

[10] Henning Kagermann. «Change Through Digitization—Value Creation in the Age of Industry 4.0». In: *Management of Permanent Change*. Ed. by Horst Albach, Heribert Meffert, Andreas Pinkwart, and Ralf Reichwald. Springer Books. Springer, Mar. 2015. Chap. 2, pp. 23–45. DOI: 10.1007/978-3-658-05014-6. URL: https://ideas.repec.org/h/spr/sprchp/978-3-658-05014-6_2.html (cit. on p. 5).

[11] Arianna Martinelli, Andrea Mina, and Massimo Moggi. «The enabling technologies of industry 4.0: examining the seeds of the fourth industrial revolution». In: *Industrial and Corporate Change* 30.1 (Jan. 2021), pp. 161–188. ISSN: 0960-6491. DOI: 10.1093/icc/dtaa060. eprint: https://academic.oup.com/icc/article-pdf/30/1/161/38822465/dtaa060.pdf. URL: https://doi.org/10.1093/icc/dtaa060 (cit. on p. 6).

[12] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010 (cit. on p. 8).

[13] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2018. 504 pp. ISBN: 978-0-262-03940-6 (cit. on p. 8).

[14] Iqbal H. Sarker. «Machine Learning: Algorithms, Real-World Applications and Research Directions». In: *SN Computer Science* 2.3 (Mar. 2021). DOI: 10.1007/s42979-021-00592-x. URL: https://doi.org/10.1007/s42979-021-00592-x (cit. on p. 8).

[15] Batta Mahesh. «Machine Learning Algorithms -A Review». In: (Jan. 2019). DOI: 10.21275/ART20203995 (cit. on p. 8).

[16] Issam Hammad, Kamal El-Sankary, and Jason Gu. «A Comparative Study on Machine Learning Algorithms for the Control of a Wall Following Robot». In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, Dec. 2019. DOI: 10.1109/robio49542.2019.8961836. URL: https://doi.org/10.1109%2Frobio49542.2019.8961836 (cit. on p. 8).

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on p. 8).

[18] Michael Grieves. «Origins of the Digital Twin Concept». In: (Aug. 2016). DOI: 10.13140/RG.2.2.26367.61609 (cit. on p. 9).

[19] Ilkka Donoghue, Lea Hannola, Jorma Papinniemi, and Aki Mikkola. «The Benefits and Impact of Digital Twins in Product Development Phase of PLM». In: *Product Lifecycle Management to Support Industry 4.0*. Ed. by Paolo Chiabert, Abdelaziz Bouras, Frédéric Noël, and José Ríos. Cham: Springer International Publishing, 2018, pp. 432–441. ISBN: 978-3-030-01614-2 (cit. on p. 9).

[20] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. «Digital Twin: Enabling Technologies, Challenges and Open Research». In: *IEEE Access* 8 (2020), pp. 108952–108971. DOI: 10.1109/ACCESS.2020.2998358 (cit. on p. 10).

[21] URL: http://www.lifetime-reliability.com/free-articles/maintenance-management/Evolution_of_Maintenance_Practices.pdf (cit. on p. 10).

[22] URL: https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/ (cit. on p. 10).

[23] Liliane Pintelon, Alejandro Parodi-Herz, Liliane Pintelon, and Alejandro Parodi-Herz. «2 Maintenance : An Evolutionary Perspective». In: 2008 (cit. on p. 11).

[24] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*. 2019. DOI: 10.48550/ARXIV.1912.07383. URL: https://arxiv.org/abs/1912.07383 (cit. on p. 11).

[25] Fatih Camci and Ratna Babu Chinnam. «Health-State Estimation and Prognostics in Machining Processes». In: *IEEE Transactions on Automation Science and Engineering* 7.3 (July 2010), pp. 581–597. DOI: 10.1109/tase.2009.2038170. URL: https://doi.org/10.1109/tase.2009.2038170 (cit. on p. 11).

[26] Omer Faruk Eker, Fatih Camci, Adem Guclu, Halis Yilboga, Mehmet Sevkli, and Saim Baskan. «A Simple State-Based Prognostic Model for Railway Turnout Systems». In: *IEEE Transactions on Industrial Electronics* 58.5 (May 2011), pp. 1718–1726. DOI: 10.1109/tie.2010.2051399. URL: https://doi.org/10.1109/tie.2010.2051399 (cit. on pp. 11, 12).

[27] J.Z. Sikorska, M. Hodkiewicz, and L. Ma. «Prognostic modelling options for remaining useful life estimation by industry». In: *Mechanical Systems and Signal Processing* 25.5 (2011), pp. 1803–1836. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2010.11.018. URL: https://www.sciencedirect.com/science/article/pii/S0888327010004218 (cit. on p. 11).

[28] An Dawn, Nam Ho Kim, and Joo-Ho Choi. «Options for Prognostics Methods: A Review of Data-Driven and Physics- Based Prognostics». In: 2013 (cit. on p. 11).

[29] Adrian Cubillo, Suresh Perinpanayagam, and Manuel Esperon-Miguez. «A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery». In: *Advances in Mechanical Engineering* 8.8 (2016), p. 1687814016664660. DOI: 10.1177/1687814016664660. URL: https://doi.org/10.1177/1687814016664660 (cit. on p. 11).

[30] Adrian Cubillo, Suresh Perinpanayagam, and Manuel Esperon-Miguez. «A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery». In: *Advances in Mechanical Engineering* 8.8 (2016), p. 1687814016664660. DOI: 10.1177/1687814016664660. URL: https://doi.org/10.1177/1687814016664660 (cit. on p. 12).

[31] P. Aivaliotis, Z. Arkouli, K. Georgoulias, and S. Makris. «Degradation curves integration in physics-based models: Towards the predictive maintenance of industrial robots». In: *Robotics and Computer-Integrated Manufacturing* 71 (2021), p. 102177. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102177. URL: https://www.sciencedirect.com/science/article/pii/S0736584521000600 (cit. on p. 12).

[32] Linxia Liao and Felix Köttig. «Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction». In: *IEEE Transactions on Reliability* 63.1 (2014), pp. 191–207. DOI: 10.1109/TR.2014.2299152 (cit. on pp. 12, 13).

[33] Huiguo Zhang, Rui Kang, and Michael G. Pecht. «A hybrid prognostics and health management approach for condition-based maintenance». In: *2009 IEEE International Conference on Industrial Engineering and Engineering Management* (2009), pp. 1165–1169 (cit. on p. 12).

[34] Jian Zhang, Tadanobu Sato, and Susumu Iai. «Support vector regression for on-line health monitoring of large-scale structures». In: *Structural Safety* 28.4 (2006), pp. 392–406. ISSN: 0167-4730. DOI: https://doi.org/10.1016/j.strusafe.2005.12.001. URL: https://www.sciencedirect.com/science/article/pii/S0167473005000585 (cit. on p. 12).

[35] Davide Mattera and Simon Haykin. «Support Vector Machines for Dynamic Reconstruction of a Chaotic System». In: *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA, USA: MIT Press, 1999, pp. 211–241. ISBN: 0262194163 (cit. on p. 12).

[36] U Thissen, R van Brakel, A.P de Weijer, W.J Melssen, and L.M.C Buydens. «Using support vector machines for time series prediction». In: *Chemometrics and Intelligent Laboratory Systems* 69.1 (2003), pp. 35–49. ISSN: 0169-7439. DOI: https://doi.org/10.1016/S0169-7439(03)00111-4. URL: https://www.sciencedirect.com/science/article/pii/S0169743903001114 (cit. on p. 12).

[37] Wei Wu, Jingtao Hu, and Jilong Zhang. «Prognostics of Machine Health Condition using an Improved ARIMA-based Prediction method». In: *2007 2nd IEEE Conference on Industrial Electronics and Applications*. 2007, pp. 1062–1067. DOI: 10.1109/ICIEA.2007.4318571 (cit. on p. 12).

[38] B. Saha, K. Goebel, S. Poll, and J. Christophersen. «Prognostics Methods for Battery Health Monitoring Using a Bayesian Framework». In: *IEEE Transactions on Instrumentation and Measurement* 58.2 (Feb. 2009), pp. 291–296. DOI: 10.1109/tim.2008.2005965. URL: https://doi.org/10.1109/tim.2008.2005965 (cit. on p. 12).

[39] O. F. Eker and F. Camci. «State-Based Prognostics with State Duration Information». In: *Quality and Reliability Engineering International* 29.4 (Apr. 2012), pp. 465–476. DOI: 10.1002/qre.1393. URL: https://doi.org/10.1002/qre.1393 (cit. on p. 12).

[40] Adem Guclu, Halis Yilboga, Ömer Eker, and Fatih Camci. «Classification of Uncertain Data Streams Using Modified K-Nearest Neighbor Method: A Case Study on Railway Turnouts». In: June 2010 (cit. on p. 12).

[41] Ronay ak, Yan-Fu Li, Valeria Vitelli, and Enrico Zio. «A Genetic Algorithm and Neural Network Technique for Predicting Wind Power under Uncertainty». In: vol. 33. Sept. 2013, pp. 925–930. ISBN: 978-88-95608-24-2. DOI: 10.3303/CET1333155 (cit. on p. 13).

[42] Runqing Huang, Lifeng Xi, Xinglin Li, C. Richard Liu, Hai Qiu, and Jay Lee. «Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods». In: *Mechanical Systems and Signal Processing* 21.1 (2007), pp. 193–207. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2005.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0888327005002165 (cit. on p. 13).

[43] Zhigang Tian. «An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring». In: *Journal of Intelligent Manufacturing* 23.2 (Nov. 2009), pp. 227–237. DOI: 10.1007/s10845-009-0356-9. URL: https://doi.org/10.1007/s10845-009-0356-9 (cit. on p. 13).

[44] URL: https://www.historyofinformation.com/detail.php?entryid=782 (cit. on p. 14).

[45] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0 (cit. on p. 14).

[46] Kunihiko Fukushima. «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position». In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. DOI: 10.1007/bf00344251. URL: https://doi.org/10.1007/bf00344251 (cit. on p. 14).

[47] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on pp. 14, 16).

[48] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661 (cit. on p. 14).

[49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 14).

[50] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. DOI: 10.48550/ARXIV.1609.04747. URL: https://arxiv.org/abs/1609.04747 (cit. on p. 14).

[51] Klaus-Robert Müller, Alexander Smola, Gunnar Rätsch, Bernhard Schölkopf, J. Kohlmorgen, and V. Vapnik. «Using support vector machines for time series prediction». In: Jan. 1999, pp. 243–253. DOI: 10.1515/9783110915990.1 (cit. on p. 15).

[52] Julia Gastinger, Sébastien Nicolas, Dušica Stepić, Mischa Schmidt, and Anett Schülke. *A study on Ensemble Learning for Time Series Forecasting and the need for Meta-Learning*. 2021. DOI: 10.48550/ARXIV.2104.11475. URL: https://arxiv.org/abs/2104.11475 (cit. on p. 15).

[53] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, Jan. 1957 (cit. on p. 15).

[54] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. 2021. DOI: 10.48550/ARXIV.2109.14545. URL: https://arxiv.org/abs/2109.14545 (cit. on p. 15).

[55] J. Brownlee. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018. URL: https://books.google.it/books?id=o5qnDwAAQBAJ (cit. on pp. 15, 16).

[56] Feng Liu, Chai Quek, and Geok Ng. «Neural network model for time series prediction by reinforcement learning». In: vol. 2. Jan. 2005, 809–814 vol. 2. ISBN: 0-7803-9048-2. DOI: 10.1109/IJCNN.2005.1555956 (cit. on p. 15).

[57] Bohdan Pavlyshenko. «Sales Time Series Analytics Using Deep Q-Learning». In: (Jan. 2022) (cit. on p. 15).

[58] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. «Deep learning for time series classification: a review». In: *Data Mining and Knowledge Discovery* 33.4 (Mar. 2019), pp. 917–963. DOI: 10.1007/s10618-019-00619-1. URL: https://doi.org/10.1007/s10618-019-00619-1 (cit. on p. 15).

[59] URL: https://towardsdatascience.com/time-series-classification-with-deep-learning-d238f0147d6f (cit. on pp. 15, 16).

[60] Kang Wang, Kenli Li, Liqian Zhou, Yikun Hu, Zhongyao Cheng, Jing Liu, and Cen Chen. «Multiple convolutional neural networks for multivariate time series prediction». In: *Neurocomputing* 360 (Sept. 2019), pp. 107–119. DOI: 10.1016/j.neucom.2019.05.023. URL: https://doi.org/10.1016/j.neucom.2019.05.023 (cit. on p. 15).

[61] Aji Prasetya Wibawa, Agung Bella Putra Utama, Hakkun Elmunsyah, Utomo Pujianto, Felix Andika Dwiyanto, and Leonel Hernandez. «Time-series analysis with smoothed Convolutional Neural Network». In: *Journal of Big Data* 9.1 (Apr. 2022). DOI: 10.1186/s40537-022-00599-y. URL: https://doi.org/10.1186/s40537-022-00599-y (cit. on p. 15).

[62] Chenyang Wang, Wanlu Jiang, Xukang Yang, and Shuqing Zhang. «RUL Prediction of Rolling Bearings Based on a DCAE and CNN». In: *Applied Sciences* 11.23 (2021). ISSN: 2076-3417. DOI: 10.3390/app112311516. URL: https://www.mdpi.com/2076-3417/11/23/11516 (cit. on p. 15).

[63] Boyuan Yang, Ruonan Liu, and Enrico Zio. «Remaining Useful Life Prediction Based on a Double-Convolutional Neural Network Architecture». In: *IEEE Transactions on Industrial Electronics* 66.12 (Dec. 2019), pp. 9521–9530. DOI: 10.1109/TIE.2019.2924605. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-02432604 (cit. on p. 15).

[64] Van Hiep Phung and Eun Joo Rhee. «A Deep Learning Approach for Classification of Cloud Image Patches on Small Datasets». In: *Journal of information and communication convergence engineering* 3.3 (Sept. 2018). DOI: 10.6109/jicce.2018.16.3.173. URL: http://dx.doi.org/10.6109/jicce.2018.16.3.173 (cit. on p. 16).

[65] Xuan Hien Le, Hung Ho, Giha Lee, and Sungho Jung. «Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting». In: *Water* 11 (July 2019), p. 1387. DOI: 10.3390/w11071387 (cit. on pp. 16, 17).

[66] Haoran Yan, Yi Qin, Sheng Xiang, Yi Wang, and Haizhou Chen. «Long-term gear life prediction based on ordered neurons LSTM neural networks». In: *Measurement* 165 (2020), p. 108205. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2020.108205. URL: https://www.sciencedirect.com/science/article/pii/S0263224120307430 (cit. on p. 17).

[67] Abdeltif Boujamza and Saad Lissane Elhaq. «Attention-based LSTM for Remaining Useful Life Estimation of Aircraft Engines». In: *IFAC-PapersOnLine* 55.12 (2022). 14th IFAC Workshop on Adaptive and Learning Control Systems ALCOS 2022, pp. 450–455. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2022.07.353. URL: https://www.sciencedirect.com/science/article/pii/S2405896322007546 (cit. on p. 17).

[68] Hao Li, Zhuojian Wang, and Zhe Li. «An enhanced CNN-LSTM remaining useful life prediction model for aircraft engine with attention mechanism». In: *PeerJ Computer Science* 8 (Aug. 2022), e1084. DOI: 10.7717/peerj-cs.1084. URL: https://doi.org/10.7717/peerj-cs.1084 (cit. on p. 17).

[69] Kwok Tai Chui, Brij B. Gupta, and Pandian Vasant. «A Genetic Algorithm Optimized RNN-LSTM Model for Remaining Useful Life Prediction of Turbofan Engine». In: *Electronics* 10.3 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10030285. URL: https://www.mdpi.com/2079-9292/10/3/285 (cit. on p. 17).

[70] Songlin Nie, Qingtong Liu, Hui Ji, Ruidong Hong, and Shuang Nie. «Integration of ARIMA and LSTM Models for Remaining Useful Life Prediction of a Water Hydraulic High-Speed On/Off Valve». In: *Applied Sciences* 12.16 (2022). ISSN: 2076-3417. DOI: 10.3390/app12168071. URL: https://www.mdpi.com/2076-3417/12/16/8071 (cit. on p. 17).

[71] Wei Huang, Hamed Khorasgani, Chetan Gupta, Ahmed Farahat, and Shuai Zheng. «Remaining Useful Life Estimation for Systems with Abrupt Failures». In: *Annual Conference of the PHM Society* 10 (Sept. 2018). DOI: 10.36001/phmconf.2018.v10i1.590 (cit. on p. 17).

[72] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. «Implicit Neural Representations with Periodic Activation Functions». In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. URL: https://proceedings.neurips.cc/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf (cit. on pp. 17, 28).

[73] H. S. Hota, Richa Handa, and Akhilesh Kumar Shrivas. «Time Series Data Prediction Using Sliding Window Based RBF Neural Network». In: 2017 (cit. on p. 27).

[74] Abien Fred Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2018. DOI: 10.48550/ARXIV.1803.08375. URL: https://arxiv.org/abs/1803.08375 (cit. on p. 28).

[75] George Casella and Roger L Berger. *Statistical Inference*. Thomson Press, Nov. 2006 (cit. on p. 29).

[76] Zhilu Zhang and Mert R. Sabuncu. *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*. 2018. DOI: 10.48550/ARXIV.1805.07836. URL: https://arxiv.org/abs/1805.07836 (cit. on p. 31).