Lehrstuhl für Maschinenelemente
Forschungsstelle für Zahnräder und Getriebesysteme
Prof. Dr.-Ing. K. Stahl

Semesterarbeit
Nr. 1888

Advanced Confidence Analysis of the State of Damage of Machine Elements Utilizing Machine Learning

Erweiterte Konfidenzbetrachtung des Schädigungszustands von Maschinenelementen mittels Methoden des Maschinellen Lernens

Eingereicht von:	Jan Nalivaika, B.Sc.
	Matr.-Nr.: 03694590
Betreuer/Themensteller:	Prof. Dr.-Ing. Karsten Stahl
	Lehrstuhl für Maschinenelemente
Beginn:	19.04.2023
Eingereicht am:	19.09.2023 in Garching bei München

Aufgabenstellung der Semesterarbeit

für Herrn Jan Nalivaika, B.Sc.

Thema:

Advanced Confidence Analysis of the State of Damage of Machine Elements Utilizing Machine Learning

Aufgabenstellung:

Eine Vielzahl von Maschinenelementen ist derart dimensioniert, dass diese unter den Belastungen des vorgesehenen Anwendungsfalls eine endliche Lebensdauer aufweisen. Durch diese Herangehensweise können Masse und Volumen der Maschinenelemente minimiert und deren Herstellungskosten und CO2 – Fußabdruck reduziert werden. Im Betrieb sind Maschinenelemente, wie zum Beispiel Zahnräder, oftmals keiner konstanten Belastung ausgesetzt, sondern deren Intensität variiert über die Betriebszeit. Um dies bei der Auslegung und der Nachrechnung zu berücksichtigen wird eine Schadensakkumulation durchgeführt. Aktuell ist hierbei der lineare Ansatz zur Schadensakkumulation am weitesten verbreitet und das durch internationale Normen vorgeschlagene Verfahren. Dieser Ansatz zeichnet sich insbesondere durch seine Simplizität und allgemeingültige Anwendbarkeit aus. Jedoch werden im Rahmen des Ansatzes eine Vielzahl von potenziellen Einflussgrößen auf die Lebensdauer bzw. den Schädigungszustand vernachlässigt, unter anderem die Reihenfolge von variierenden Belastungen. Dies kann zu einer verminderten Konfidenz der Ergebnisse und damit zu einer unzuverlässigeren Prognose der Restlebensdauer führen. Der Einfluss der Reihenfolge der Belastung ist analytisch sehr komplex zu erfassen, da mit einer enormen Varianz an möglichen Verläufen umgegangen werden muss. Deshalb soll im Rahmen dieser Studienarbeit eine Möglichkeit erarbeitet werden Methoden des Maschinellen Lernens mit dem Ziel einzusetzen, dem Anwender eine Bewertung des Konfidenzbereichs einer bestimmten Restlebensdauer zu erleichtern. Basis der Bestimmung der Restlebensdauer ist immer zunächst der aktuelle Schädigungszustand, deshalb soll sich diese Arbeit auf die Bewertung von diesem beschränken.

Hierbei sind insbesondere folgende Arbeitspakete zu bearbeiten:

- Einarbeitung in die Grundlagen des Maschinellen Lernens und der Betriebsfestigkeit
- Sichtung und Aufbereitung von bereitgestellten Versuchsdaten zur Lebensdauer von Zahnrädern
- Systematischer Entwurf eines Konzepts basierend auf Methoden des Maschinellen Lernens um die Konfidenz von linearen Schädigungszustandsberechnungen zu bewerten

- Aufbau eines Programms, das ausgehend von Belastungs-Zeit-Verläufen die Zuverlässigkeit von Schädigungszustandsberechnungen einschätzt (zum Beispiel mittels der Programmiersprache Python und geeigneter Bibliotheken)
- Aufbereitung und Einordnung der erzielten Ergebnisse
- Übersichtliche und vollständige Dokumentation des Stands der Technik, der Arbeitsschritte und der Ergebnisse

Die Arbeit wird nach ihrer erfolgreichen Fertigstellung in der Lehrstuhlbibliothek veröffentlicht. Informationen, Daten und EDV-Programme, die dem Studenten während der Bearbeitung dieser Arbeit vom betreuenden Assistenten oder anderen Mitarbeitern des Lehrstuhls zugänglich gemacht werden, sind als streng vertraulich zu behandeln und verbleiben Eigentum des Lehrstuhls. Die Nutzung ist ausschließlich für die Bearbeitung und Erstellung dieser Studienarbeit gestattet.

Die Nutzungsrechte an dieser Arbeit gehen an den Lehrstuhl für Maschinenelemente, Forschungsstelle für Zahnräder und Getriebesysteme, der Technischen Universität München über. Dem Verfasser wird ein privates Nutzungsrecht gewährt.

Garching bei München, den 19.04.2023

Jan Nalivaika, B.Sc.

Daniel Vietze, M.Sc.

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorgelegte Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Garching bei München, den 19.09.2023

Jan Nalivaika, B.Sc.

Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer Daniel Vietze, vom Lehrstuhl für Maschinenelemente an der Technischen Universität München, bedanken. Ohne seine Bereitschaft und kontinuierlichen Unterstützung während des gesamten Prozesses wäre diese Arbeit nicht entstanden.

Table of contents

1 Introduction

1.1 Motivation

As “Industrie 4.0” became the new standard, more and more data is continuously gathered in every industry process [Vog16]. By analyzing this large amount of data, new options are discovered and implemented that can help save resources, time, and money [The15]. The use-cases range from yield improvement in chemical processes, condition monitoring in machinery, to fraud detection in banking [Jan21]. New insight from that data enables more specialized solutions and significantly faster improvements in almost all processes [Stu10].

To analyze the gathered data, more and more sophisticated methods have been developed over the last few years. Not only did the available computing power take a significant leap forward, but also the methods for data analysis and knowledge extrapolation, which are now widely available [Jan21]. One of the most prominent methods is Machine Learning (ML) which is part of Artificial Intelligence (AI) [Hel20]. Since the early beginnings of AI in 1956, the idea of ML has gotten more and more public attention [Stu10]. Today, ML is applied in all areas where traditional algorithms cannot handle the amount of data or the problems are too abstract [Car19a]. Image recognition and chatbots are just a small example of the possibilities ML is able to offer [The15].

Especially engineering sciences are interested in the implementation of ML and the associated improvement of processes [Car19a]. Saving money by optimizing processes and workflows is one of the biggest driving forces for ML. In more and more use cases, like predictive maintenance and anomaly detection, ML is the new normal [The15].

Gears are one of the most common elements in mechanical engineering and can be found in almost every part of industry applications [Vul20]. Those gears are dimensioned, manufactured, and treated in such a way as to require the least amount of material and still fulfill their purpose without failure throughout their desired lifetime [Nic13]. This thesis takes a deeper look at how the application of ML can support the confidence in the calculated current state-of-damage by specifically considering the time history of the loads applied onto the gear.

1.2 Problem and Goal Formulation

1.2.1 Problem Statement

In mechanical engineering, machine elements are usually designed to withstand loads in an expected range over their desired course of life [Vie20]. Those loads are mostly not static but dynamic and can vary significantly in their magnitude over the operating time [Wit17]. For example, gears in a car’s transmission or gearbox experience a variety of loads depending on the torque output of the engine [Yuk04]. Rapid acceleration with a heavy trailer has a different time-load profile than just cruising at constant speed.

To determine the current state-of-damage, multiple methods can be employed [Lee11]. One of the most used methods is the Miner rule [Min45] (also called the Palmgren-Miner rule) which is one option of the linear approaches [Sun14]. The Miner rule is also referred to in international standards [ISO19].

One of its main advantages is its simplicity and general applicability. The Miner rule assumes that each load cycle is responsible for a certain percentage of damage. Higher loads are responsible for more damage. The accumulated damage for one load level is proportional to its applied cycles. A machine element is expected to fail as soon as the accumulated damage sum D reaches 1.0 (also referred to as unity) [ISO19, Mil77].

When comparing this approach to experiments, it is shown that the Miner rule does not always correctly calculate the true physical state-of-damage [Pav18]. For example, multiple experiments were conducted with gears that showed that the damage sum D, calculated with the Miner rule, is not always a reliable parameter to determine the failure point [Han01].

These different points of failure across the applied load cycle history are not due to the natural and expected variance in the strength of the machine elements but are related to the order of applied loads [Sko99]. Depending on the order (history) of loads that are applied, gears are failing earlier than expected (before the damage sum accumulates to unity) or are capable of handling significantly more load cycles, even though the calculated accumulated damage sum exceeds 1.0 [Han01].

The Miner rule completely neglects to take the variability of the order of loads into account. For that approach, it is irrelevant when a load is applied [ISO19]. This leads to reduced confidence in the results and is thus an unreliable prediction of the remaining service life and current state-of-damage. The influence of the sequence of loads is very complex to capture in a simple model since, for a correct calculation, all previous cycles and loads need to be taken into account [Vie20]. Even though this specific linear model has an established and proven history in engineering, its limitations can be improved upon for a correct state-of-damage calculation.

There are other models that are based on non-linear approaches but are very complex and tedious to use [Vie20]. Further, the general applicability is not given, as some of these approaches rely on constants that are specific to the material of the analyzed machine element [San18].

When looking at gears specifically, the production process needs to be taken into consideration. As high-quality gears are first machined and then post-processed with heat treatment and steel ball blasting (shot peening), internal stresses are introduced that can affect the gears' ability to handle more load in a certain range than expected [Ben02]. Thus, not all theories are easily transferable from one machine element to another. The current state of the art does not offer a method that can reliably predict the remaining useful life (RUL) and current state-of-damage in machine elements. The prediction of the RUL is especially difficult, as the future loads on a machine element are highly uncertain.

1.2.2 Aim of this Thesis

As the development of a new fatigue analysis model is out of scope for this thesis, a different approach is selected. The aim of this thesis is to develop a method, utilizing ML, that can analyze the state-of-damage in machine elements based on the time-load history and return a confidence indicator of the calculated damage sum that was acquired using a linear damage-accumulation method.

In practice, this method helps machine operators, who have already implemented methods for state-of-damage calculations, get a better understanding of how long they can expect their machines to operate. By using this method, unexpected early failures can be avoided and unnecessary maintenance stoppages reduced.

The individual subgoals of this thesis are: the pre-processing of the available data; the conceptual development of the ML-method; the implementation of the method in a suitable programming language; and finally, the analysis and documentation of the results.

1.3 Structure of the Thesis

Chapter 2 gives an introduction to the topic of fatigue analysis and ML. Special attention is given to the application of ML in industrial applications. Chapter 3 presents a method that can return a confidence value for the calculated state-of-damage based on a given load history. All the required pre-processing and implementation steps are also discussed. Chapter 4 focuses on the analysis of the results. Finally, chapter 5 summarizes the results and gives an outlook on future work.

2 State of Research and Development

The following chapter gives a brief introduction to the topic of fatigue analysis in mechanical engineering. Further, the topic of ML and its core functionalities is presented. Finally, it is analyzed how ML is applied in industry, especially in fatigue analysis of machine elements.

2.1 Introduction to Fatigue Life Analysis

The goal of fatigue analysis is the dimensioning of machine elements in such a way that they are able to withstand the expected loads over an expected time period [Hai06]. The strength of a machine element depends on the used material as well as the geometry [Wit17].

2.1.1 Load Sequence and Load Spectrum

The stresses resulting from repeated loading of a mechanical component can be plotted over time. This loading over time is referred to as load–time history. When omitting the time component and only keeping the order, it is referred to as the loading sequence [HEU05]. In this thesis, the load–time history and loading sequence are treated as equal. In cases where the amplitude of the load in the loading sequence is not constant for each load cycle, the process is referred to as variable amplitude loading [Fac18]. Figure 2.1 shows a repeated loading with a constant and variable amplitude. The resulting stresses are reaching a constant or varying level, depending on the load pattern. Depending on the area of application, the y-axes can also show a moment applied to a component like a driveshaft.

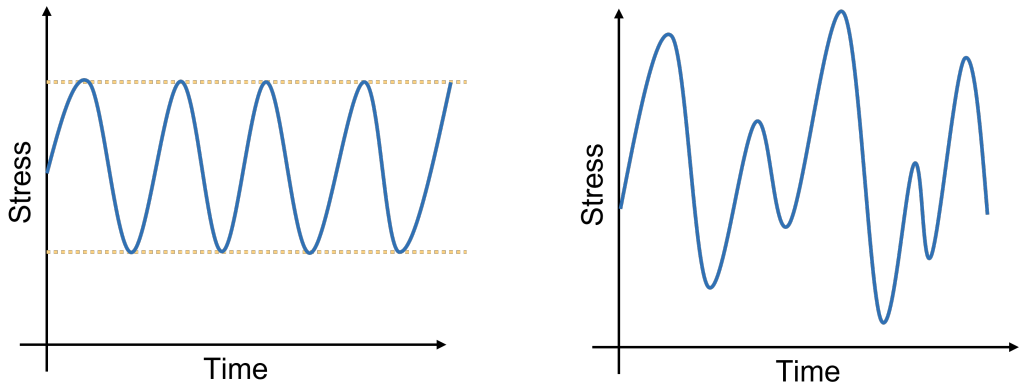

Figure 2.1: Constant and variable amplitude loading

The Load Spectrum deals with the occurrence of load cycles with different amplitudes within a load sequence [Fac18]. To get a load spectrum from a load sequence, the time and order component are omitted. A load spectrum shows the number of occurrences of a certain load amplitude in a load sequence. By eliminating the order and time components of a sequence, the same load spectrum may arise from multiple different sequences. Depending on the desired detail in the load spectrum, the load levels of a sequence can also be rounded. Depending on the area of application, the rounding can be done to the next integer or the next 10,000.

The rounding will increase the occurrences at certain load levels as more occurrences are projected in each category. Figure 2.2 shows a schematic load spectrum calculated from an exemplary load sequence.

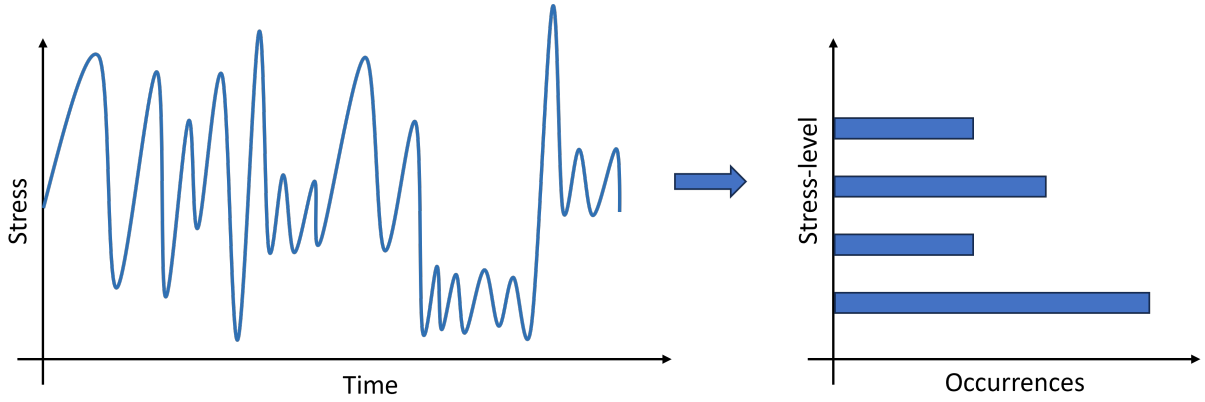

Figure 2.2: Load Spectrum from Load Sequence

2.1.2 The S-N Curve

The S-N curve (also referred to as the Wöhler-curve) plays a very important role in fatigue analysis. It is applied in all aspects of engineering [Bur18, Pun06]. The graphical representation gives clear insights into the maximum number of load cycles a material is capable of withstanding before a fatigue break is expected. The applied load (S) is placed on the y-axes and the number of cycles (N) on the x-axes. The curve shows the number of cycles to fatigue failure and is individual for every material and for every different component. The S-N curve can also be determined for elements like welded joints or whole structures [Bap17, Don08]. Both axes are on a logarithmic scale to achieve two straight lines passing from the upper left to the lower right of the graph [Lit81]. Figure 2.3 shows a schematic representation of the S-N curve.

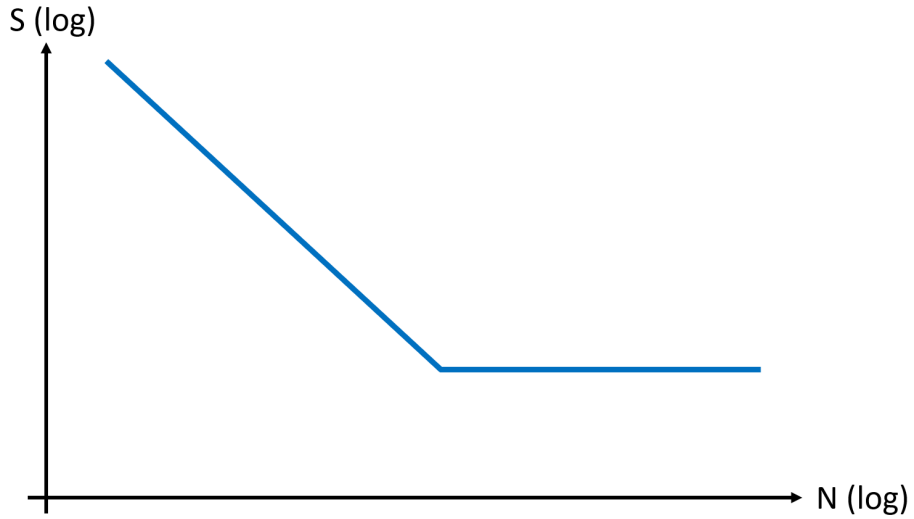

Figure 2.3: Schematic representation of a S-N curve

As can be seen in figure 2.3, the curve is made up of two straight lines. The first part, which has a negative slope, is referred to as a fatigue strength range. In that range of loads, the material shows a limited number of cycles before failure occurs [Ada20].

The horizontal line marks the endurance limit, also known as the fatigue limit. Below that load level, it is assumed that a material can withstand an infinite number of cycles without failure [Bel22]. It is often denoted as the S-N curve plateau. It is important to note that the S-N curve is determined by experiments that are conducted in controlled environments. Thus, they are not always representative of real-life machinery in industrial applications. Real-world applications may involve additional factors, such as varying load levels, environmental conditions, and surface finishes, which can influence fatigue performance [Jan82].

Figure 2.4 gives a clear insight on how the S-N curve is actually a probability distribution and not a fixed deterministic line. This distribution arises from the different number of cycles achieved in experiments at the same stress level before failure occurred. Depending on the component and material, the resulting distribution is not necessarily a Gaussian distribution.

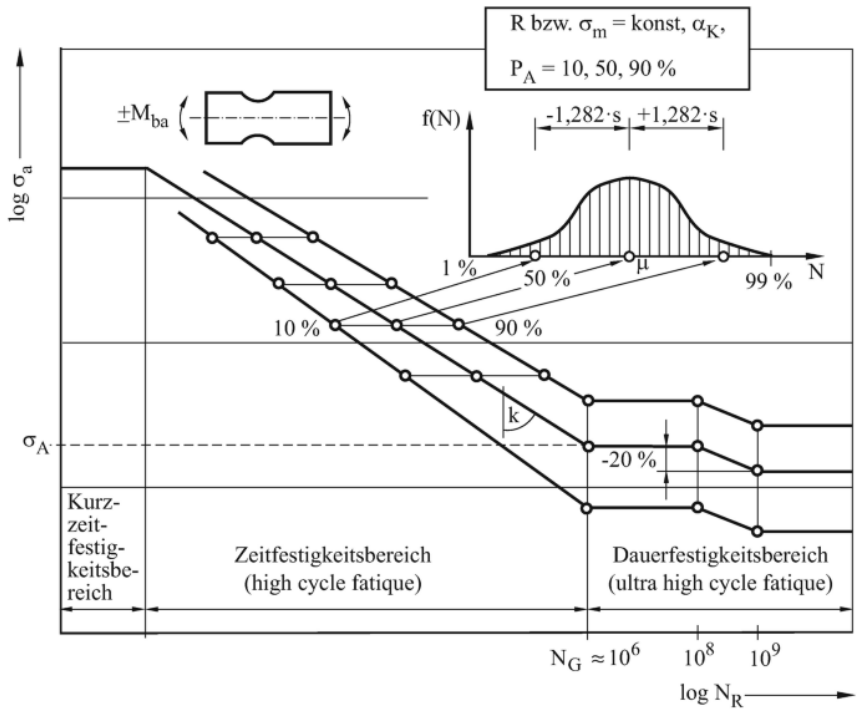

Figure 2.4: S-N as a distribution [Kle19]

2.1.3 Linear Damage Accumulation Method

The S-N curve is a crucial component in the application of the Miner rule [Min45, Sub76]. The Miner rule is a popular linear damage accumulation technique most commonly used in engineering. It is based on three assumptions. First, that the rate of damage accumulation is constant for each loading cycle of one load level. Second, damage only occurs if the applied stress is higher than the fatigue limit (S-N curve plateau). This assumption is only true for the Miner-original and not for Miner-elementar or Miner-Haibach [Wer00]. And third, the component is expected to fail

when cumulative damage reaches the unity [Zuo15]. The Miner rule also assumes that the first load cycle at a certain stress level is just as damaging as any other at that stress level. The order of loads is not relevant when calculating the accumulated damage.

Each load cycle damages the machine element inversely proportionate to the number of maximum loads the component could withstand at that stress level. For example, if a component is capable of withstanding 1,000 loads at a certain stress level, one of those cycles adds 1/1000 to the accumulated damage of the machine element. If another component can withstand 50 cycles at a different stress level, each of those cycles adds 1/50 to the accumulated damage.

For varying loads, the sum of those individual ratios forms the accumulated damage. For example, 25 percent of the fatigue life is used up if a part is repeatedly loaded for 500 cycles at a stress level that would lead to failure in 4,000 cycles and then followed by 2,000 cycles at a stress level that would lead to failure in 16,000 cycles. Equation 2.1 is how the accumulated damage sum D is calculated. At stress level S_1, S_2 ... S_i, the corresponding numbers of load are n_1, n_2 ... n_i. The number of cycles to failure under each stress level are N_1, N_2 ... N_i.

$$D = \sum_{j=1}^{k} \frac{n_i}{N_i} \tag{2.1}$$

As mentioned before in chapter 2.1.2, the individual points N_i in the fatigue strength range are determined by applying one load in a cyclic manner, to determine the maximum acceptable cycles. When using this number as a ground truth in the Miner rule, the effects of other loads, happening before or after, are neglected. To solve this, non-linear damage accumulation methods were developed.

2.1.4 Non-Linear Damage Accumulation Methods

One of the first non-linear fatigue damage accumulation models was the expansion of miner rule by a power law [Zuo15]. Equation 2.2 shows how the damage sum D is calculated. The parameter C_i is a material parameter corresponding to the i-th loading level.

$$D = \sum_{j=1}^{k} \left[\frac{n_i}{N_i} \right]^{C_i} \tag{2.2}$$

The disadvantage of this method is that the parameter C_i must be calculated for different loading conditions and is thus tedious to acquire [Zuo15].

Another approach, used in [Reg17], proposes to use iso-damage-curves as a basis to calculate the cumulative fatigue damage.
Many more approaches can be found in [Che20, Gao14, Lv15, Zhu19]. The common disadvantage of non-linear models is that they are dependent on very individual and specific parameters, limited to certain materials or loading histories, with a fixed trend regarding the loading amplitudes. Due to those limitations and cumbersome usage, they are not widely adopted in the engineering field [Vie20].

2.2 Overview of Machine Learning

In contrast to rigid algorithms, ML is a method that can solve a given problem without being explicitly told how to solve it [Jan21, Sut20]. The following chapter discusses the advantages, general functionality, and core principles of ML.

2.2.1 General Introduction to Machine Learning

ML is a subsection of the field of general AI [Hel20]. The same as deep learning (DL) is a subsection of ML [LeC15]. AI encompasses the general principles and methods that can mimic human behavior and decision-making [Jan21]. ML, on the other hand, incorporates all methods that can learn from data and discover regularity and patterns that are not obvious to a human [The15]. Based on the discovered knowledge, the trained model can be used to make predictions for new inputs in the future. DL is a subset of ML that refers to methods that utilize a special approach called a Neural Network (NN). The "depth" in DL in this context refers to the amount of stacked layers of neurons (see chapter 2.2.5) [Car19a]. DL is more capable than shallow ML and shows improved learning capabilities, but it also comes with its own specific challenges [Jan21, LeC15]. Figure 2.5 shows how DL and ML are subsets of AI.

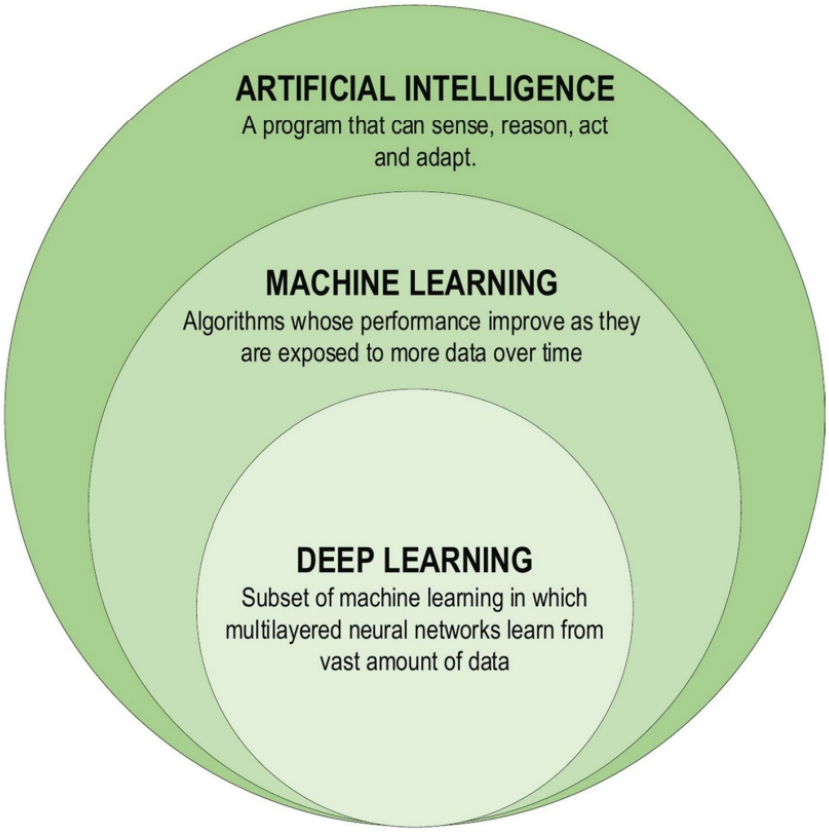

Figure 2.5: Venn-Diagram of the different subsets of AI [Alz21]

2.2.2 Advantages and Problems of Machine Learning

Advantages of Machine Learning

The most important advantage of ML is the ability to learn from data and discover the underlying, non-obvious patterns [Wue16]. With that knowledge, it is also possible to predict the output for a specific input. A ML model deployed to a self-driving car could, for example, predict how a pedestrian is going to cross a street and calculate if the car should perform an emergency brake to avoid a collision. One further advantage of ML is its wide applicability [Tho22]. Not only computer science or the automotive industry, but also banking, healthcare, and many more fields, can all find applications where ML can help improve a process.

These algorithms are also very efficient and profit greatly from more available data [Jan21]. Not only is it possible to learn from a lot of data, but also to process a lot of data [Wue16]. One example is the analysis of sensor data, for example, from LiDAR sensors. Those sensors produce a lot of data that might need to be analyzed as quickly as possible, in some cases even in real-time. For self-driving cars, fast analysis of this data is crucial for avoiding accidents. ML is capable of solving this problem reliably [Lyu19].

The era of "Big Data" plays right into the hands of ML. More data results in better performance of the ML approaches [Wue16]. Also, the methods can be specially adapted to be robust. Outliers, noise, and missing values in the data do not skew the results in a significant way [The15]. When comparing ML to DL, ML can tend to produce better results in cases where the training data is limited and also allows for a better interpretation of the outputs of the model [Jan21].

Disadvantages of Machine Learning

The disadvantages of ML are closely related to the advantages. For example, the data-set that is used for training needs to be a certain minimum size, so the model is able to learn the underlying patterns. But not only the size is important, but also the contained elements, as some models can return different results based on the elements that make up the data-set. The effectiveness of ML rises and falls with the amount and quality of the data [Bis06, Jan21].

One of the biggest problems of ML is generalization [Bis06]. Finding and learning patterns is just the first step, but making a correct prediction is the most valuable part. Depending on the free parameters of ML models, they have the capability to memorize the whole presented training-set and perform very poorly on data that is not part of the training-set [Zha17]. This case is called "over-fitting" [Jab14]. In such cases, the model does not learn the underlying structure of that data but just the correct associations from one input to one output.

The opposite of "over-fitting" is called "under-fitting" [Jab14]. The model is not capable of learning any patterns, and thus performs very badly on the training set and on newly presented test samples. This happens mostly when the data-set is not large enough for the chosen model or the model is not flexible enough to adapt to the pattern of the data [Wil18]. In a worst-case scenario, a ML method could interpret the noise of the data as a pattern and perform very poorly when applied in real-world applications.

Especially for high-dimensional data and complex problems, a very large dateset is a must. This can be difficult to obtain, especially in the area of natural sciences or industry [Wue16]. Sensors can provide very corrupt data, and some experiments can take a lot of time and money to perform. And if that data is acquired, it still requires a significant amount of manual work to bring it into a form where ML methods can work with it. To train a model, for example, to classify multiple bacteria cells, it first takes time to grow the bacteria, but then also to label all the pictures. For example, the MNIST data-set for number recognition contains 70,000 elements, that were all manually labeled [Pav17]. So depending on the area or application, it is impossible to obtain a data-set that has the necessary size and quality.

Further, models can be very susceptible to hyperparameters [Jan21]. Hyperparameters are fixed values that describe the characteristics of a model. Those values are set manually and require extensive knowledge about the implemented model, the problem at hand, and the available data-set [Luo16]. The optimal parameters can be found with an extensive brute force approach or specialized algorithms, but they require too much time to be feasible in an industry-application [Cla15].

Table 2.1 summarizes the advantages and disadvantages of ML. Despite its disadvantages, the abilities and unique advantages of ML make it a very interesting addition to industry applications [Ber21]. Especially as more and more capable methods are developed. The main goal when implementing ML is to work around the disadvantages to achieve the desired results.

Advantages of ML	Disadvantages of ML
• can learn from data	• requires large datasets
• can make predictions based on learned patterns	• data is tedious to acquire
• versatile in applicability	• is sensitive to hyperparameters
• can process a lot of data efficiently	• can be sensitive to training-set
• can handle noise and outliers in data	• over-fitting / under-fitting

Table 2.1: Advantages and Disadvantages of ML

2.2.3 General Terms in Machine Learning

In the following, some of the general terms in ML are explained. As the field of ML is very large and continuously growing, this section focuses only on the core elements. Understanding these terms and principles helps to get a better understanding of the ML methods and how they can be applied in industry. For a more in-depth look, more information can be found in extensive literature [Bis06, Goo23, The15].

Training and Testing

To train and test the model's ability to generalize, the data is split up into a training-set and a test-set. The training-set is used to determine the optimal parameters of the model [Bis06]. The process of finding the optimal parameters is referred to as training or learning. It is important to keep both sets separate and only use them for their respective purposes. The training-set must not contain elements of the test-set and vice versa. Otherwise, if the expected error is computed

on the training-set, the ability of the model to generalize is not correctly assessed [Xia16].

One easy way to understand the search for optimal parameters during training is with the example of curve-fitting of a polynomial function (see [Bis06] for a more in-depth look).
Function 2.3 is a polynomial of degree M with free parameters w_i that can be adapted to fit the data as accurately as possible. The error function, defined in equation 2.4, also called loss function, measures the accuracy of the polynomial. The parameters are changed in such a way to reduce the error. If the polynomial is flexible enough, the error is reduced to zero. In the case of ten free parameters and ten data points (x_n,t_n) the polynomial passes through every data point exactly. The error in that case is zero.

$$y(x,w) = \sum_{j=0}^{M} w_i j^j = w_0 + w_1 * x^1 + w_2 * x^2 + ... + w_M * x^M \tag{2.3}$$

$$E(w) = \frac{1}{2} \sum_{n=0}^{N} \{y(x_n,w) - t_n\}^2 \tag{2.4}$$

Figure 2.6 shows how the polynomial changes based on the number of free parameters, where M is the degree of the polynomial. When the polynomial, displayed in red, reaches degree nine, the error is zero. In this case, however, it is not following the true data generation function shown in green and performs very poorly on new unseen data points (see "over-fitting" in chapter 2.2.2).

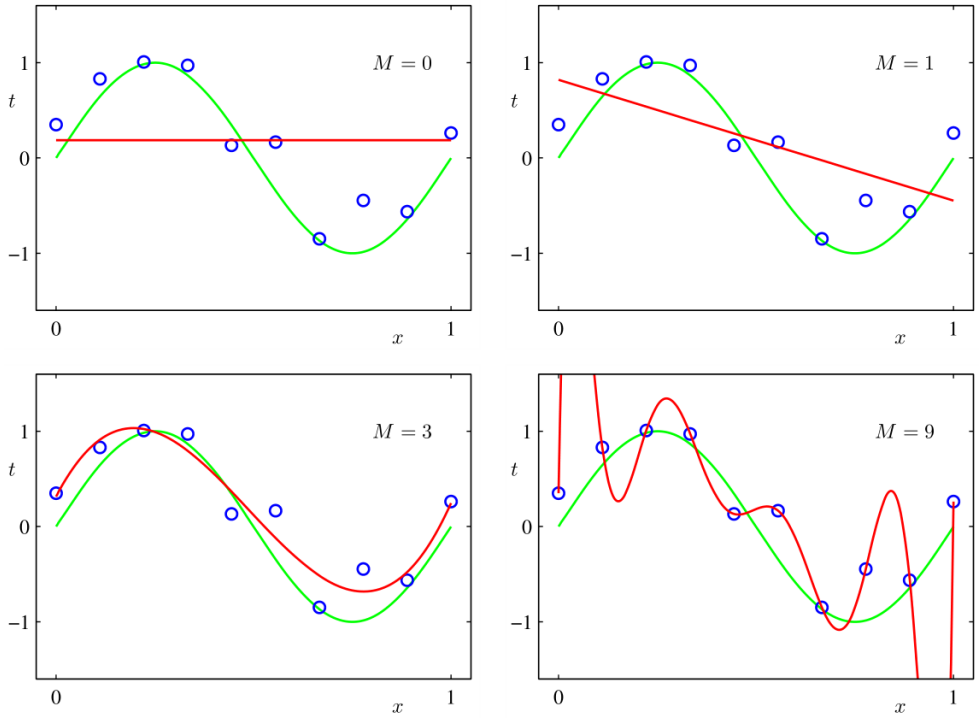

Figure 2.6: Influence of the number of free parameters in curve-fitting [Bis06]

Features and Feature Selection

Every model needs an input and a corresponding output. As computers work on a digital principle, physical values have to be transformed into a digital format. For example, sensor data from analog pressure valves or analog current meters has to be transformed into digital values. "Industrie 4.0" did a lot of work to replace standalone analog sensors with sensors having a digital output or wireless connection, so that this translation step is no longer a hurdle. The post-processing of those signals can be done on a central computer or dedicated edge devices on the shop floor [Lu23].

But not all data is accepted by a ML model right away, even if it is in digital form. These methods need an input that is specifically shaped for each specific model. When the goal is to apply a ML model for image classification, the image needs to be transformed into a format that is accepted by the implemented model. Images can have different height-to-width ratios (e.g., 16:9 or 4:3) and different resolutions (e.g., 1920x1080 pixel or 64x64 pixel). All the images need to be transformed into a uniform shape with the same resolution [Par20].

For example, if an image data-set contains mostly images with a 1:1 ratio, wider images need to be contracted or cut off to fit the pattern so that all images are the same. The resolution must also be adjusted so that all images have the same number of pixels.

Figure 2.7 shows how images of different sizes and resolutions are adapted to a fixed resolution and ratio. In the upper image, the overhanging parts on the sides are removed, and no resolution scaling is necessary. In the lower image, parts on the top and bottom are removed. Additionally, the resolution is downscaled from 125x125 pixel to 100x100 pixel.

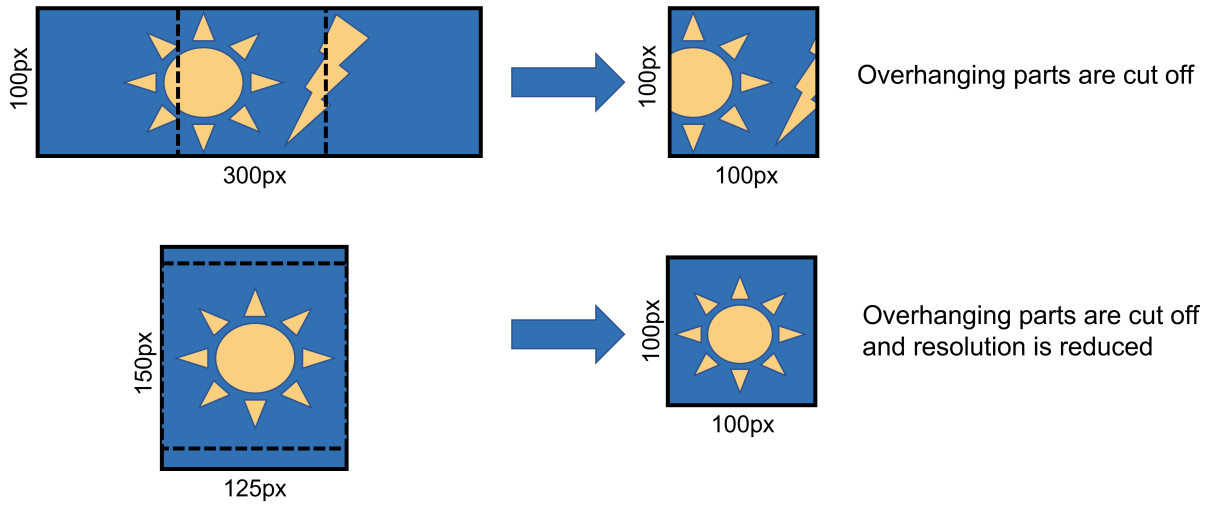

Figure 2.7: Changing resolution and ratio of images for a homogeneous data-set

The ratio and resolution depend on the implemented method and can be chosen freely. The current standard for images (MNIST) that are used in bench-marking of various algorithms is a squared format with a 28x28 pixel resolution [Bal19]. For optimal classification, a higher resolution proves advantageous but can require longer training time and more complex models (e.g. a larger NN) [Hua20, Kan18]. After the input elements are standardized, they can be transformed into a vector or matrix format. The process of constructing such designed input from raw input is called "encoding". The result after the encoding is called a "feature vector". The elements of such a vector are referred to as features. One set of features uniquely represents the uncompressed input [The15]. It is possible to reduce the feature vector through a process called "feature selection", where the feature vector is reduced even further to a smaller size that contains the gist of the raw input. By reducing the dimension, the models are capable of learning faster, but this process is running the risk of omitting valuable data in the process [Jan21].

Figure 2.8 shows how a matrix, representing an image, for example, can be encoded as a vector. As long as the encoding method is constant for all input elements, the order is irrelevant. In that case, the elements can also be stacked not only row-by-row but also column-by-column.

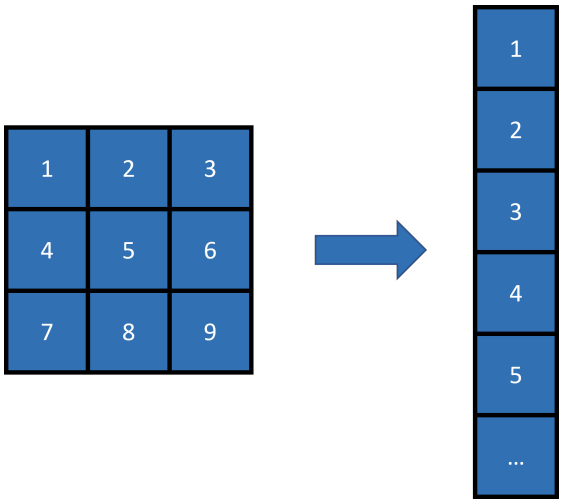

Figure 2.8: Encoding of an matrix as a vector

Data augmentation

As mentioned in chapter 2.2.2, a large data-set improves the success of ML approaches. Instead of inflating a data-sets with noisy or corrupt samples and relying on the algorithm's ability to handle such data, a different approach is possible. The process of enlarging a given data-set is called Data Augmentation (DA). The goal is to bypass the problem of small datasets and increase their size and quality [Sho19]. By doing so, over-fitting can be prevented. DA is mostly used on images, but can also be transferred to all other types of data. Possible DA approaches are, for example, geometric transformations like cropping, flipping, scaling, and rotating [Tay18]. Further possibilities are changing the contrast, brightness, white-balance, tint of the image, as well as local pixel manipulations [Mik18]. These individual adjustments can also be combined to produce a plethora of data.

Further approaches are possible when working with data in a time-series format [Ban21, Wen22]. One definition of time-series, is data that has a time component, like recorded sensor data, and thus can be plotted over time [Ham20]. Another definition is that the data has an orderly element and not necessarily a time component [Iwa21]. An example of a series without a time component are words in sentences. In such cases, cropping might distort the data too much to be representative of the original element.

One of the most straightforward approaches when working with signal data is to overlay the data with Gaussian noise or any other noise pattern. Another option is inserting or emitting repeating values of the data, which makes the time-series longer or shorter [Wen22].

Figure 2.9 shows further possible DA techniques. The different methods can be divided into basic and advanced approaches [Wen22].

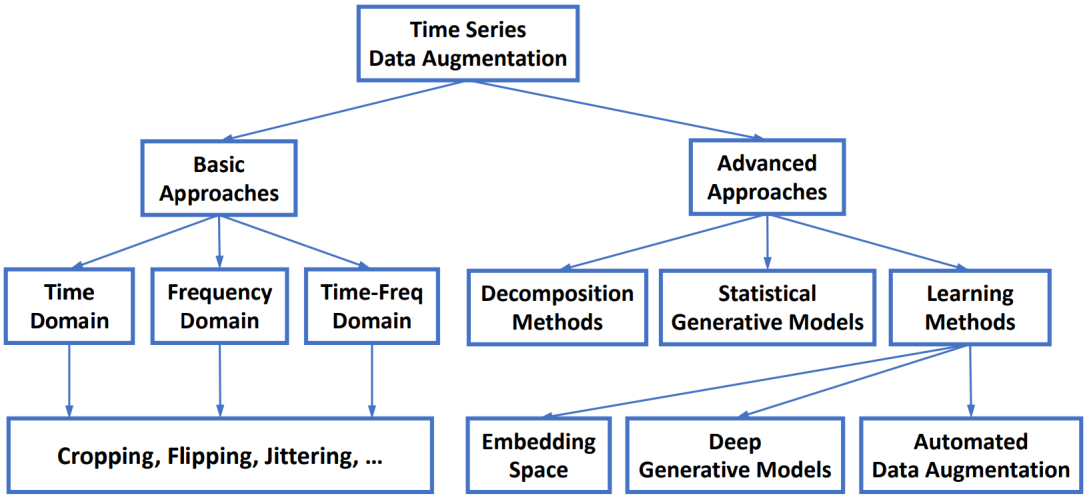

Figure 2.9: Possible DA techniques for time-series data [Wen22]

Figure 2.10 shows how a polynomial can be varied to increase the data-set size but still keep its significant characteristics.

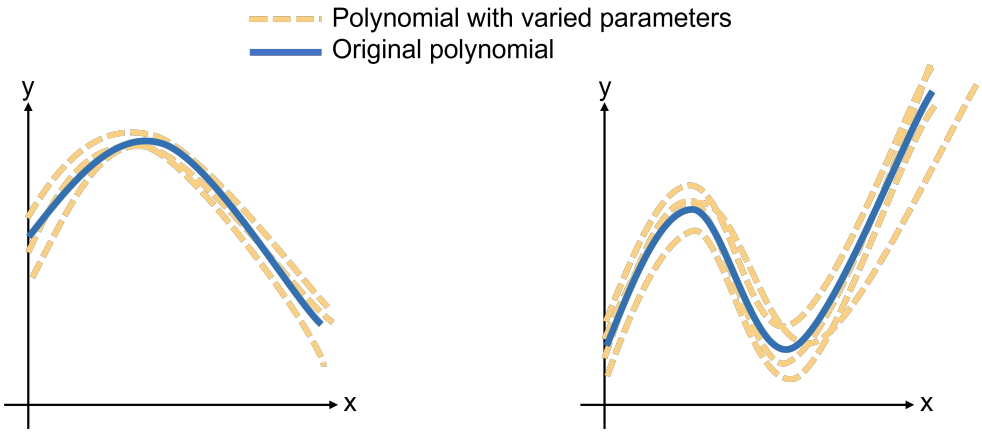

Figure 2.10: Application of DA to polynomial functions

Data Pre-Processing

When working with a given data-set, it is helpful to the ML method if all feature vectors are in the same range across all samples. The aim is that the greater numeric feature values are not dominating the smaller numeric feature values, and each one can have an equal contribution to the output. This improves the performance of ML-methods significantly [Sin20]. The process of changing the data is referred to as pre-processing. In the following, a few selected data pre-processing methods are discussed in detail.

The first and one of the most important pre-processing steps is the removal of outliers [Yan19]. Outliers can be values that do not fit the range of expected values or have no entries (missing data). In some circumstances, these data points are easy to filter out, as they can be compared to the real-world processes where that data was recorded. For example, a temperature recording from a melting furnace is expected to be in the range of 600 to 1000 degrees Celsius. Negative and close-to-zero values can be easily removed, as these values are not physically possible during operation. Removing outliers reduces the variance of the training set and is responsible for a significant performance boost [Li15].

Normalization (also called Min-Max scaling or Min-Max Normalization,) is the process of projecting the data into a specific interval. This interval is most commonly set between -1 and 1 [Pes14]. This method of scaling produces good results only if no outliers are present in the data-set. The outliers could skew the scaling such that the non-outlier values are projected onto a very small range of the interval and are very difficult to be distinguished from one another. The task of normalization is frequent in feature engineering. When all the numerical features in the feature vector fall within approximately the same range, models often train more quickly and make better predictions [Jay11].

The standardization approach involves remapping the features by subtracting the mean and dividing by the variance. This approach is used if the data is made up of multiple sources and the elements are not on the same scale. The data is standardized to prevent characteristics with wide ranges from impacting the output compared to the other values. Data is standardized to make it center on a certain value (0 is chosen most often) with a variance of 1 [Raj20]. One example of standardization is when a future vector contains the pay and the years of employment of an employee. As the pay is numerically much larger than the years of employment, it is necessary to standardize all the elements across all feature vectors.

Further possible pre-processing options include MaxAbs-scaling, Robust Scaler-scaling, and Quantile Transformer [Ahs21]. Figure 2.11 shows how elements of a feature vector are projected in the range [-1;1] with the help of a MinMax-scaler. After the projection, numbers four and five are no longer distinguishable.

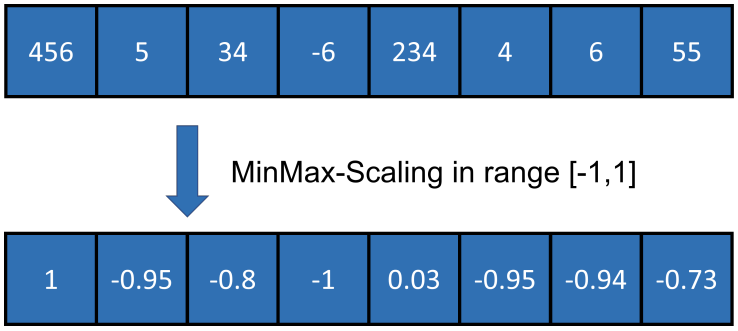

Figure 2.11: Applying a MinMax-Scaler to a feature vector

Hyperparameter

Before a ML model can be trained, a set of hyperparameters needs to be defined that characterize the model and its behavior. Most of the time, they are set manually, relying on the user's intuition or taken from literature [Phi19]. For optimal performance, it requires extensive knowledge of the selected algorithm and how different combinations of parameters affect each other. Exemplary hyperparameters are: learning rate, loss function, activation function or weights of neurons in NN (see chapter 2.2.5) [Yan20b]. As hyperparameters play a very important role in the success of a ML model, specialized algorithms are in development for optimal hyperparameter selection [Ber13].

K-fold Cross Validation

K-fold Cross Validation (KCV), is an approach for error estimation of ML models [Zha23]. By applying KCV, the data-set is split into k subsets. All but one subset are used for training and the remaining one is used to measure the model's performance after it is trained. For that, the error-function is used. The score is saved, and the process is repeated for all k subsets [Yos03]. In the end, all k subsets were used once to assess the model. The expected value over all k-iterations is the final estimation of the models' performance. By employing this process, the model's sensitivity to one particular data-set does not influence the final rating of the model's performance [Bis06].

2.2.4 Core Functionalities in Machine Learning

The core functionalities of ML are Supervised Learning, Unsupervised Learning and Reinforcement Learning (RL) [Jan21, The15]. In the following, these three functionalities are discussed in more detail. Semisupervised Learning is a mix of Supervised and Unsupervised Learning [Zhu05]. Due to its rarity and specialized approach, it is not covered by the scope of this thesis.

Supervised Learning

Supervised Learning is split up into two parts: Classification and Regression [Jan21, The15]. In the case of Classification, the data elements, that a model is trained on, have a label (class label) [Car19a]. For example, images that contain either cats or dogs. The image is the data.

The label is a corresponding value of either 1 or 0, which represents the presence of either animal in one specific image (for example, 0 for cats and 1 for dogs). The goal is to predict if a new image contains cats or dogs. The model that is used in the case of Classification is called a classifier [The15].

To return the correct label, the classifier has to train on the available trainig-set, and find correlations between the data and labels [Car19a]. After each training cycle, the prediction of the classifier can be compared to the true label (ground truth) and thus supervised on its performance. The classifier must learn the underlying structure (the pattern) of images and has to generalize, as the pictures in the test-set are not used in training and thus are unseen [Bis06]. The found patterns allow the classification of the input (image) into discrete and limited variables (type of animal, 0 or 1) [The15].

Figure 2.12 shows a schematic representation of how a classifier is trained on a data-set containing images of either dogs or cats and how it is used to predict the label of a new image.

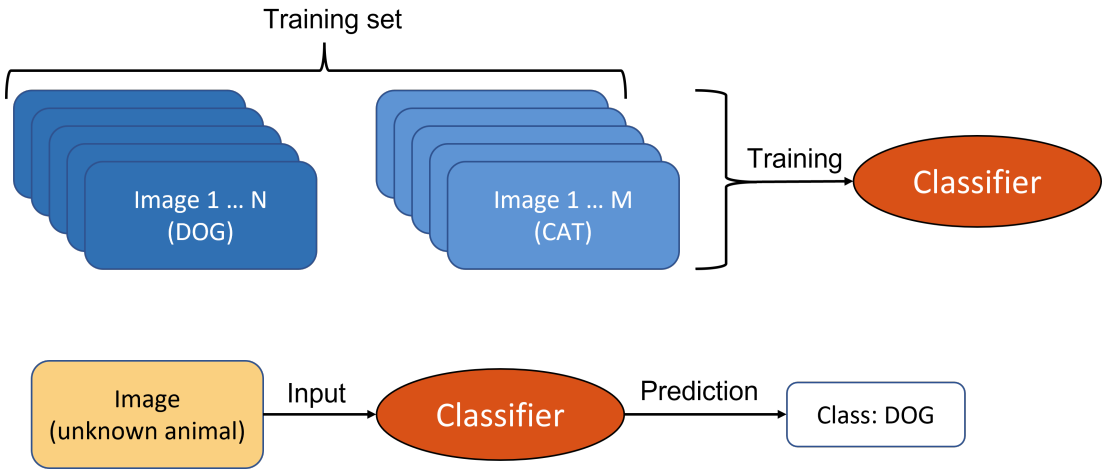

Figure 2.12: A schematic of the training and use of a classifier

Further examples of classifications are: the prediction of whether a tumor is benign or malignant or the authorship of a given text [The15].

The same basic principle is applied to regression. The main difference is that in the case of regression, the output is not a binary class but a continuous variable. In other words, the labels are lying in an interval. In broader terms, it is possible to call it a "curve fitting" method (see chapter 2.2.3) [The15].

One application, for example, is to predict the position of the sun based on a picture containing shadows. The data in this case is, just as in the classification, the image. The label, on the other hand, is now a number that is the solar azimuth angle. The model can fit a function to the available data points and predict the sun's position on a new (unseen) picture. An even more simple example is shown in figure 2.13. A given data-set contains values of points in the x-y plane. With a regression model, the mapping function is estimated. Now the estimated function can be used to predict the y-value for a new x-value.

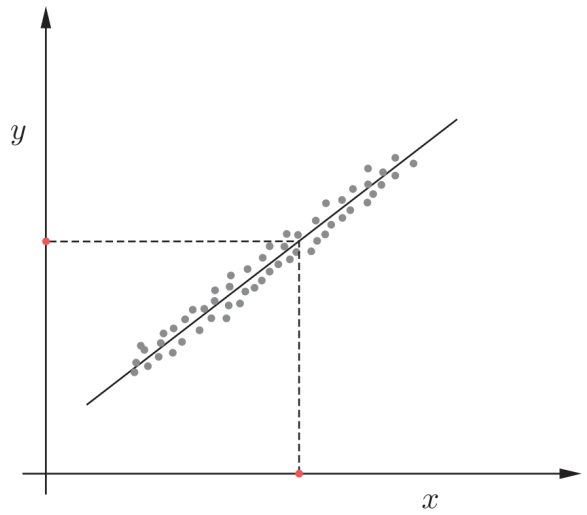

Figure 2.13: A mapping function based on a data-set containing x and y values [The15]

Another example of regression is to predict a stock price in the next few minutes based on the history of the last 30 minutes [Jan21].

Unsupervised Learning

In Unsupervised Learning, the data that is given to a model has no labels. The goal of that method (just like in Supervised Learning) is to also find underlying patterns [Car19a]. But instead of learning a pattern and comparing the output to the ground-truth, the method is designed in such a way that it discovers a pattern on its own [Mur12]. The focus is not on assigning labels to an input, but on deciding to which class it belongs.

When applying this method to images, the output of the model are groups of pictures, where every group has something in common. Unsupervised Learning is very similar to human learning, as most things are learned without someone explicitly explaining how something works [Mur12]. As mentioned in section 2.2.2, large labeled datasets are very difficult to create. Unsupervised Learning does not require a label and is thus widely applicable.

Figure 2.14 visualizes how an unsupervised learning approach can divide a set of shapes based on their geometric properties. Class 1 contains shapes that are mostly round and class 2 contains shapes that are angular.

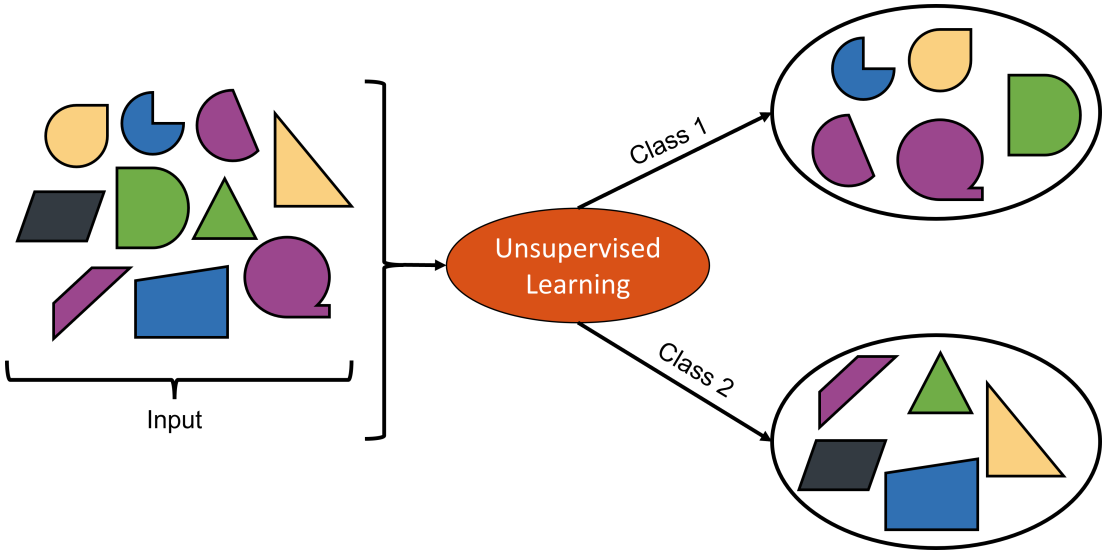

Figure 2.14: Finding similarities in shapes and grouping them into classes

Reinforcement Learning

RL is based on the same principle as the learning process in people or animals. The main objective is to reach a goal, or in other words, to maximize a numerical value. The learning process is in a loop with observing and decision-making. The so-called agent is in an environment that he can observe and perform action in. The actions are rewarded or punished with a numerical reward. [Ami18]

Actions that bring the agent closer to the goal are rewarded proportionally to their usefulness in achieving the goal. After completion of the task (achieving the goal) a large reward is issued to signal the agent that the steps taken were successful. As the agent does not know which actions lead to high rewards, it has a significant exploration part in the beginning of its training to confirm or update its assumptions regarding the predicted reward for each action in each state. [Sut20]

One of the most challenging tasks in RL is deciding which steps in the process were the good and bad ones. This is also called the "Credit Assignment Problem" [Bis06, Min61]. To solve a task, for example, finding a way out of a labyrinth, the agent has to first navigate in the opposite direction to the exit, to then again get closer to the exit. The three distinguishing parts of RL are: a closed loop system, no clear instructions, and the long-term strategy of the agent. [Sut20]

Figure 2.15 shows one iteration of the interaction of the agent with the environment. First, the agent receives the state of the environment as a feature vector. Based on that current state and the agent's long-term strategy, it performs an action and thus changes the environment. After performing the action, the agent is rewarded for its performance based on the defined goal [Ami18].

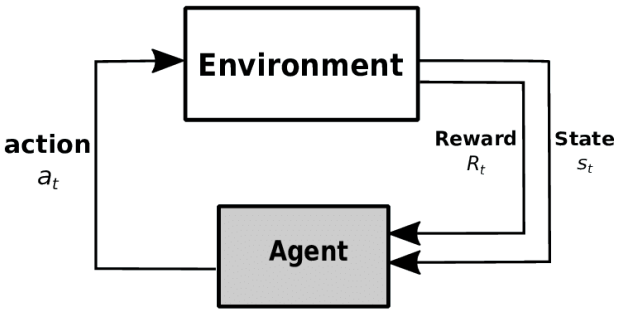

Figure 2.15: The interaction of an agent with its environment in RL [Ami18]

RL finds a lot of applications in games and game-like scenarios, as the goal and reward are easily defined and the state of the environment is easily encoded into a feature vector [Tak22]. Due to the limited scope of this thesis and the nature of the problem described in chapter 1.2.1, RL is not further discussed. More information can be found in [Sut20].

2.2.5 Specialized Machine Learning Approaches

In the following, three specialized architectures are presented and analyzed in more detail.

Neural Networks

NNs are based on the same principle as the functionality of the human brain [Jan21]. The basic components of a NN in a living organism are neurons and synapses. The synapses are connecting the neurons with each other. Based on the input, a synapse can be activated or inhibited. Based on that observation, McCulloch and Pitts implemented a simple model, called a perceptron, that mimics exactly this behavior [The15]. Figure 2.16 shows how two neurons are connected to each other. This principle is reproduced many times to create a large NN out of perceptrons.

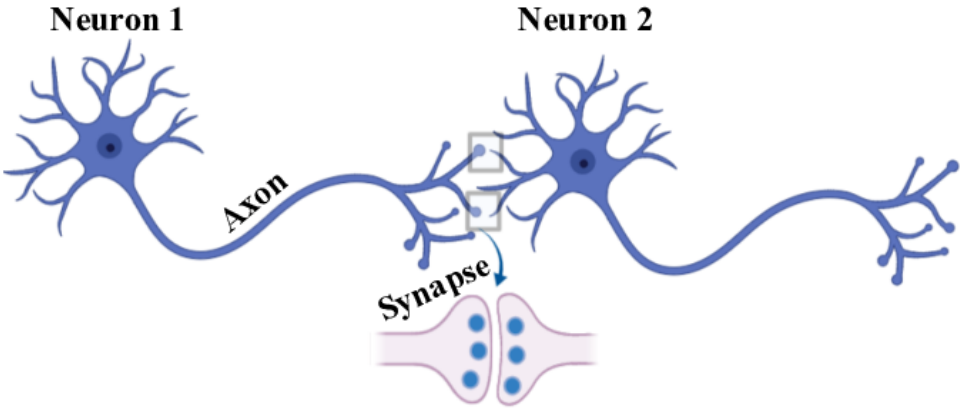

Figure 2.16: Connection of two neurons via synapses [Das22]

NN are also referred to as Artificial Neural Networks (ANNs). These models are based on the same principle as the perceptron. In this case, the perceptrons are arranged in layers and referred to as nodes or neurons. The first layer is called the input layer, and the last one is the output layer. The layers in between are called hidden layers [Ver21].

Figure 2.17 shows an exemplary NN with three hidden layers. Each element of the input vector is given to one node at the input layer. This type of NN is also called a Feed-Forward Neural Network. Because the neurons are fully connected, it is also referred to as a fully connected NN [Sai15]. Note that the output does not need to be a single value. It can also be a vector. In that case, the output layer would consist of multiple nodes.

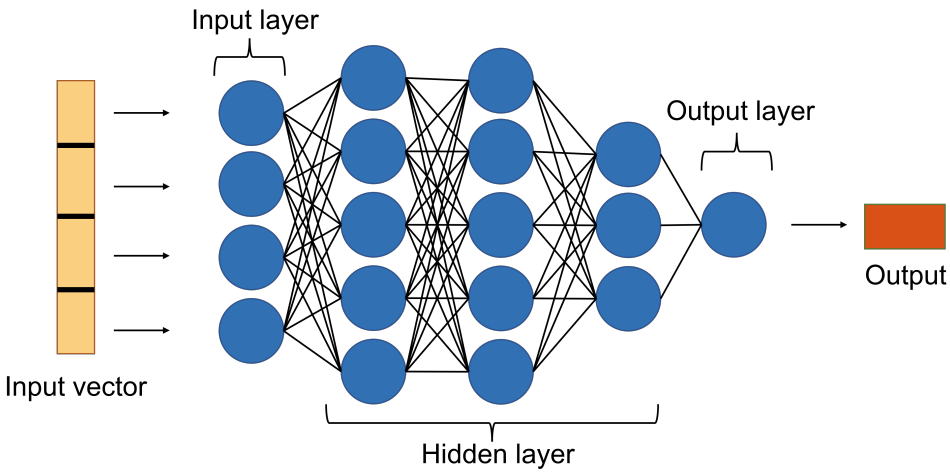

Figure 2.17: NN with three hidden layers

To model an organic NN more closely and improve the performance, the nodes have an activation function built into them, which can introduce non-linearity [Goy20]. One of the most common activation functions is the "rectified linear unit" (ReLU) [Goo16]. Figure 2.18 shows three common activation functions. The most common activation functions are: ReLU, sigmoid, tanh and softmax [Gil23].

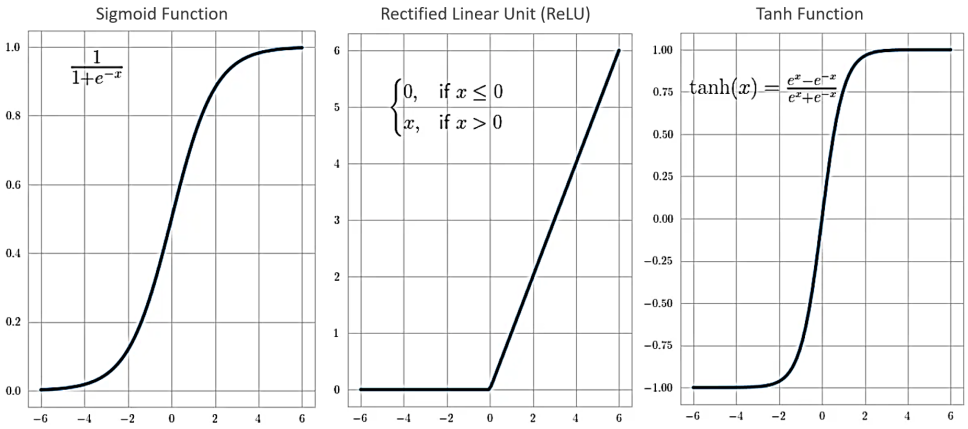

Figure 2.18: Various activation functions for NNs [Gil23]

Figure 2.19 shows the basic functionality of a neuron at the first hidden layer. The ouput-elements of the input layer are pre-multiplied with individual values called weights and summed up to a single value. This value is processed by the activation function and returns the output for that specific neuron for that specific input [Bat19]. The output value is then used as input in the following hidden layers [Din18]. The adjustments of those weights is done in training through a process called back-propagation [Cha95]. It is important to note that the activation functions can be chosen for each layer individually. Especially in classification, the activation function at the output layer is most commonly set as SoftMax or thanh while in the previous layers as ReLU [Asa20].

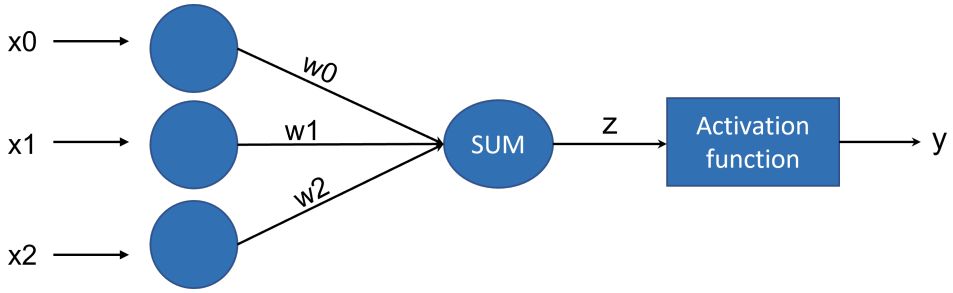

Figure 2.19: Graphical representation of a perceptron [Din18]

Deep Neural Networks (DNNs) are NNs that contain many hidden layers. The increased amount of parameters and non-linearity helps them achieve better feature abstraction and learn more complex patterns [Ver21]. The biggest disadvantage of DNNs and DL in general is that even more data is necessary to optimally tune the free parameters [Tho20]. Further, DL is very computationally expensive and requires significantly more time for training [Fu17].

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are specialized NNs that are designed to be applied to problems involving time-series data, like sensor data or ordered data like sentences [Jai99]. After each input, the neurons in a RNN are sending feedback signals to the following and some previous nodes. By doing that, a cycle is formed [Gro13]. By having this feedback loop to previous nodes, the output is dependent on the new input and the output from the previous input. This specialized architecture can pay attention to the sequence of the input data [Jai99].

The rest of the functionality is almost the same as in a conventional NN. Figure 2.20 shows a simple RNN. The output of the first layer is sent back to the nodes C1 and C2. In this case, they are called context units. With this structure, the RNN can "remember" the input from the previous input. These closed-loop cycles are the "memory" of the RNN [Sal17].

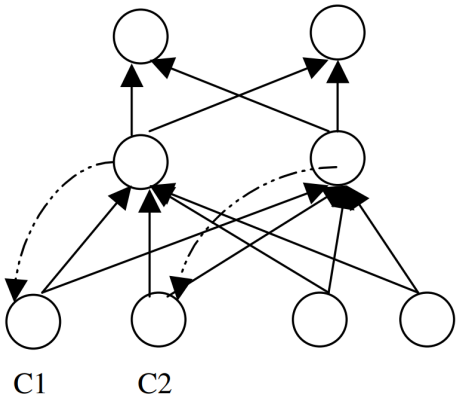

Figure 2.20: Simple RNN [Jai99]

By using the memory-structure, RNNs are capable of understanding sequential dependencies. A conventional RNN is capable of remembering up to ten time-steps (inputs) back. By adding more recurrent connections with the goal of increasing the memory-capacity, the gradient of the response is either exponentially increasing to infinity or decreasing to 0 [Sta19]. Thus, a standard RNN is limited in its ability to remember a lot of data.

Figure 2.21 shows how a RNN can be unfolded. The recurrent structure is unfolded in the time-domain and gives a clear understanding of how the output of a cell at time t_0 is used as additional input at time t_1.

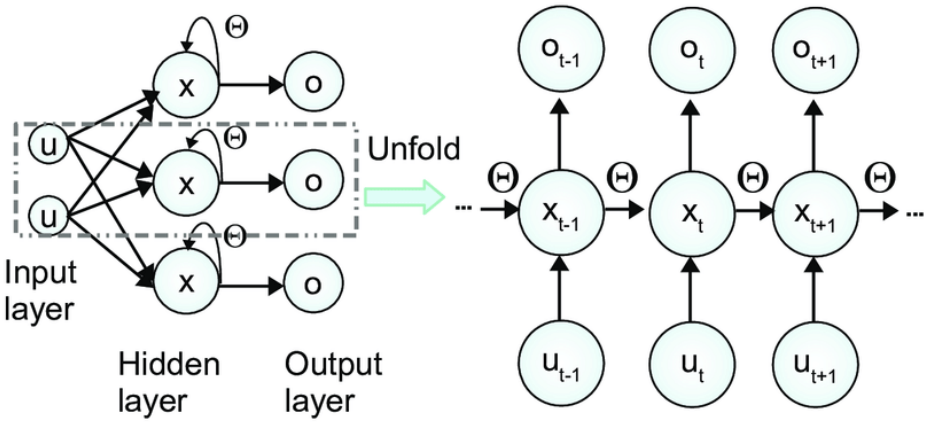

Figure 2.21: Unfolded structure of a RNN [Guo20].

A subgroup of RNNs are long-short- term memory RNNs (LSTMs). LSTMs are able to learn the correlation of even longer sequences than conventional RNNs [Sal17]. LSTMs are capable of remembering more than 1,000 time-steps [Sta19].

LSTMs are finding more and more areas of application. For example, in machine health monitoring or load prediction in power stations [Muz19, Zha16]. The similarity between those applications is the dependency of the output on recent and past inputs.

Physics Informed Neural Networks

Physics-informed Neural Networks (PINNs) are special types of NNs that are capable of following defined boundary conditions like the laws of physics [Cai21]. One of the most commonly used application domains, are problems involving partial differential equations (PDEs). Instead of using the available data to discover the physics from scratch, prior knowledge can be incorporated through the PDEs [Cuo22]. By utilizing PINNs, a simple NN architecture can be used that requires less training time, and can achieve equivalent results to large NNs [Mis20].

Figure 2.22 shows the basic functionality of a PINN. The overall goal is to minimize the error function, just as in function fitting (see chapter 2.2.3). In this case, the error function is made up of the initial- and boundary-conditions as well as the constraints of the PDE [Guo20]. The training process is analogous to the standard NN approach.

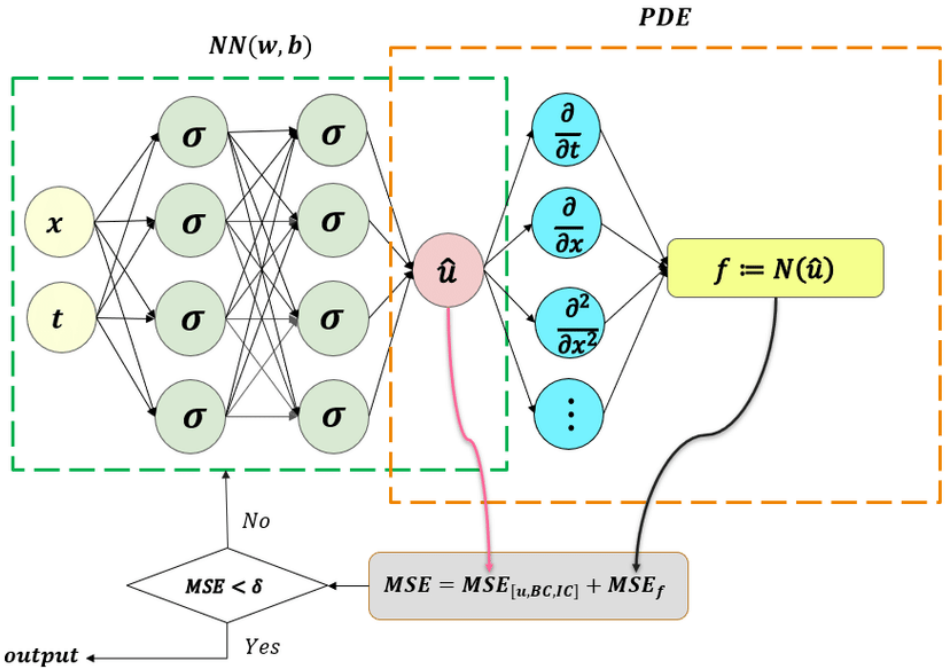

Figure 2.22: Schematic diagram of a PINN [Guo20]

2.3 Overview of Machine Learning in Industry Applications

In the following, the application of ML in engineering and industrial applications is discussed.

2.3.1 General Application of Machine Learning in Industry

The possibilities of ML are extending to many areas of application [Dei22]. The analysis of recent publications in [Ber21], concluded that there are four main categories of ML for industrial applications. Table 2.2 gives to each category an example.

Application Domain:	Example:
Maintenance Management	Failure modes classification and prediction
Quality Management	Defects detection and classification
Production Planning and Control	Job scheduling and dispatching
Supply Chain Management	Demand planning and forecasting

Table 2.2: Main categories of ML for industrial applications [Ber21]

When comparing the implemented ML-approaches, Supervised Learning clearly dominates each category. When looking at the total field, each ML approach (Supervised, Unsupervised and RL) experienced a significant growth increase over the last 20 years [Ber21]. This is a clear indication that ML will play an important role in the future for engineering sciences.

Condition Monitoring and Predictive Maintenance (which are both in the domain "Maintenance Management") are closely related to the problem explained in chapter 1.2.1. Condition monitoring is the continuous analysis of parameters that correlate to the machine's operation and maintenance [Rao96]. Predictive Maintenance is a preventive maintenance program. Instead of scheduling maintenance activities after a failure has occurred, predictive maintenance uses condition monitoring to run the machines as long as possible (close to failure) to eliminate unnecessary downtime through premature stopping or unexpected delays through unforeseen failures [Mob02].

It is important to note that Predictive Maintenance and Condition Monitoring are also performed with other tools, like statistical analysis and do not solely rely on ML [Car19b, Div23].

2.3.2 Comparison of Machine Learning in Gear Applications

The process of calculating health indicators has gained a lot of interest in engineering in the past few years. By correctly assessing a system's state, accidents and unexpected downtime can be avoided [Wan18]. In this section, the application of ML in Condition Monitoring and Predictive Maintenance of gears is analyzed.

The basis of health indicators are signals or values coming from the physical system itself. Vibration, noise, electrical current and thermal images can be used for fault detection or to determine the state-of-damage in gears [Kar20, Kar22, Med19, Pra14].

By using the vibration signal, it is possible to determine gear damage like tooth cracks and face wear. These kinds of damage can introduce vibrations that travel to other parts of the system like the gearbox housing. By having an accelerometer on the housing, the vibrations can be detected and analyzed. The proposed method in [Pra14] was able to successfully make a distinction between a faulty and a healthy gear. This method, however, did not make any predictions regarding the progressing state of health of the gears and only analyzed them in their current state-of-damage.

The approach in [Med19] uses a LSTM network to determine the progressing level of pitting. The input signal is classified into nine levels of severity of damage. The pitting damage was created

artificially by Electrical Discharging Machining (EDM). The input data is an acoustic recording of 15 seconds. In total, 1215 samples are recorded. After prepossessing and encoding, the LSTM could successfully classify each signal into the correct class of pitting damage.

The method applied in [Tiw09] also uses vibration signals to perform a classification. In this approach, a Support Vector Machine (SMV) is used to classify healthy and worn gears. The goal is to classify different gears based on their damage. Some gears had chipped or missing teeth. This approach also achieved a high success rate. No attention is given to the change in vibration over a long period of time to determine a state-of-damage.

These approaches are not transferable to the problem described in chapter 1.2.1. When the datasets for those methods are constructed, the gears are already in the state that is used as a label (healthy, damaged).

When trying to transfer this approach to get a better prediction than the Miner rule, the input would be the load-history and the label would be the true state-of-damage or range of it. But calculating the physically accurate damage sum D is not possible. Thus, these approaches are not capable of solving the problem at hand.

Over the last decade, significant advancements have also been continuously made in the area of predicting the Remaining-Useful-Life (RUL) of machine elements [Deu18, He21, Yan20a]. The biggest problem with those models and approaches is the assumed constant load that is acting on the components. Due to this assumption, these approaches can not be transferred to machinery where the cyclic loads have a significant variance in amplitude over time.

In summary, no methods were published that use a-priori knowledge of the Miner rule and the known load sequence to determine a more confident state-of-damage in machine elements.

3 Methodology

This chapter discusses the methodology that uses load sequences that ended in fatigue failure, to get a confidence value for the accumulated damage sum D of a load sequence that did not yet end in failure. Each step of the proposed method is discussed in detail. Chapters 3.1 and 3.2 give an introduction to the data, while chapter 3.3 serves as general guidance. Detailed explanations begin in chapter 3.4.1.

3.1 Data Acquisition

The load sequences are acquired by performing a Single Tooth Bending Fatigue Test (STBF). A gear is held in place by two teeth clamped in a test rig. A hydraulic piston is applying cyclical loading until bending fatigue failure occurs. The loading profile is manually defined for each test. The loading profile consists of a force level and the number of cycles during which this force is applied. Instead of constructing the loading sequence from the predetermined parameters (force and repetitions), a force sensor is used that records the net force of a cycle that occurs at its peak. This is done to simulate real machinery, where the loading history is also measured and not created by a-priori available hyperparameters. Figure 3.1 shows an exemplary test rig for STBF tests.

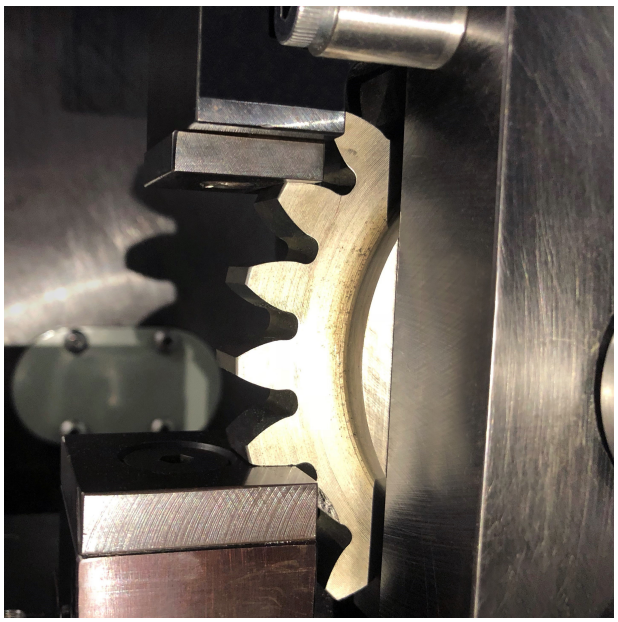

Figure 3.1: Test rig for Single Tooth Bending Fatigue Test [Sym23]

3.2 Data Visualization

Figures 3.2, 3.3 and 3.4 show three different load sequences. The x-axis denotes the number of the cycle in that sequence, and the y-axis the applied force. As the force is acting in the negative direction with regard to the defined coordinate system, it is recorded with a negative sign. The clamping force is set to 3 kN. The sudden reduction of force in the load history appears when the test rig switches from one load-level to another. Additionally, the vertical lines represent the damage sum D according to Miner at 0.5, 0.8 and 1. Note that failure is expected at a damage sum equal to 1. The tested gear with the load sequence presented in figure 3.3 was able to accumulate a damage sum D greater than unity, and is thus exceeding its expected failure point. The applied load sequence shown 3.4 resulted in early failure.

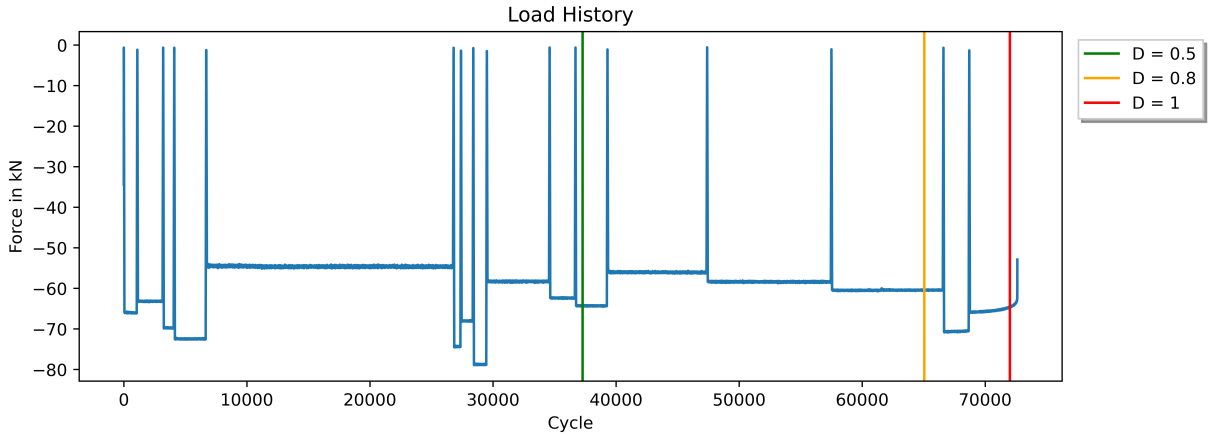

Figure 3.2: Load sequence with failure close to D=1

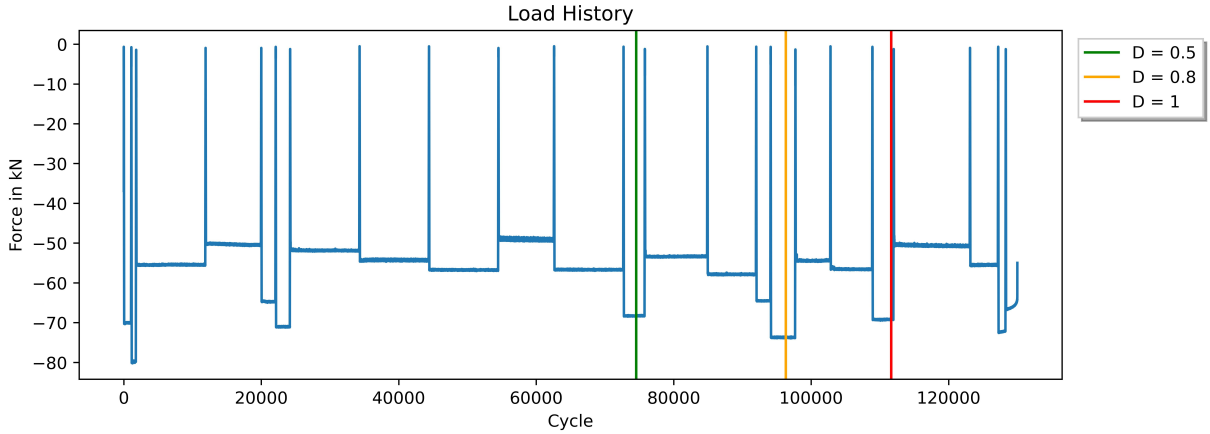

Figure 3.3: Load sequence with failure at D»1

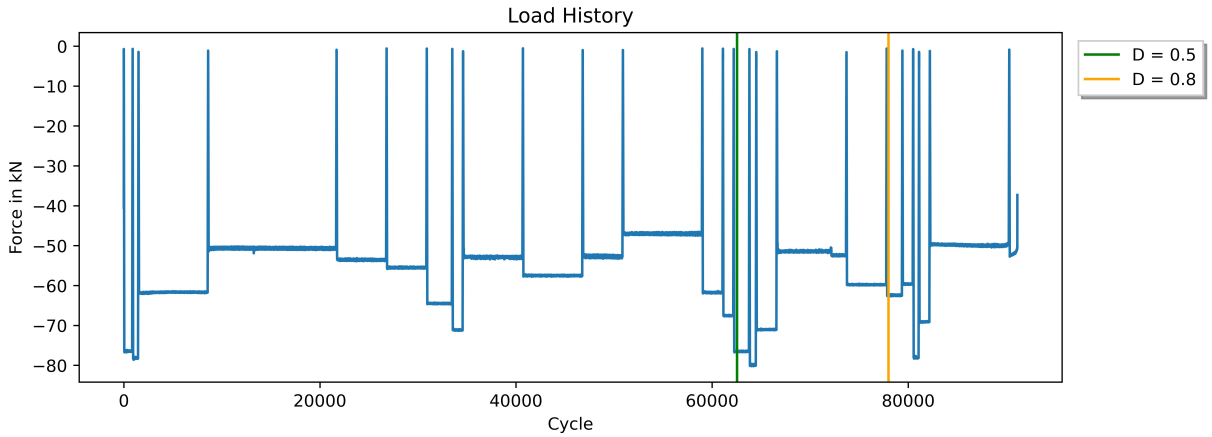

Figure 3.4: Load sequence with failure at D«1

Note that for a ML approach, the absolute values are irrelevant and can be changed as long as all load sequences are transformed in the same way. Flipping the sign or dividing by a constant to get the stress per unit area will result in equal performance of the selected method.

3.3 General Overview of the Proposed Structure

The first goal is to use a classifier to determine if a load sequence will result in early (D<1), late (D>1) or on-time failure (D≈1). In other words, if a load sequence is more or less damaging than assumed by the Miner-Rule, based on the order of the loads. Based on that knowledge, it is possible to deduce the confidence in the damage sum D that was calculated with the Miner rule. For example, if the load sequence is classified as part of the less damaging ones, it is fair to assume that the Miner rule gives a conservative estimate of the actual physical damage.

After that, a regression is performed to estimate a new defined health indicator called "State of Health" (SOH). Figure 3.5 shows the process of classifying a load sequence with unknown effects based on the available load sequences that are used as training data.

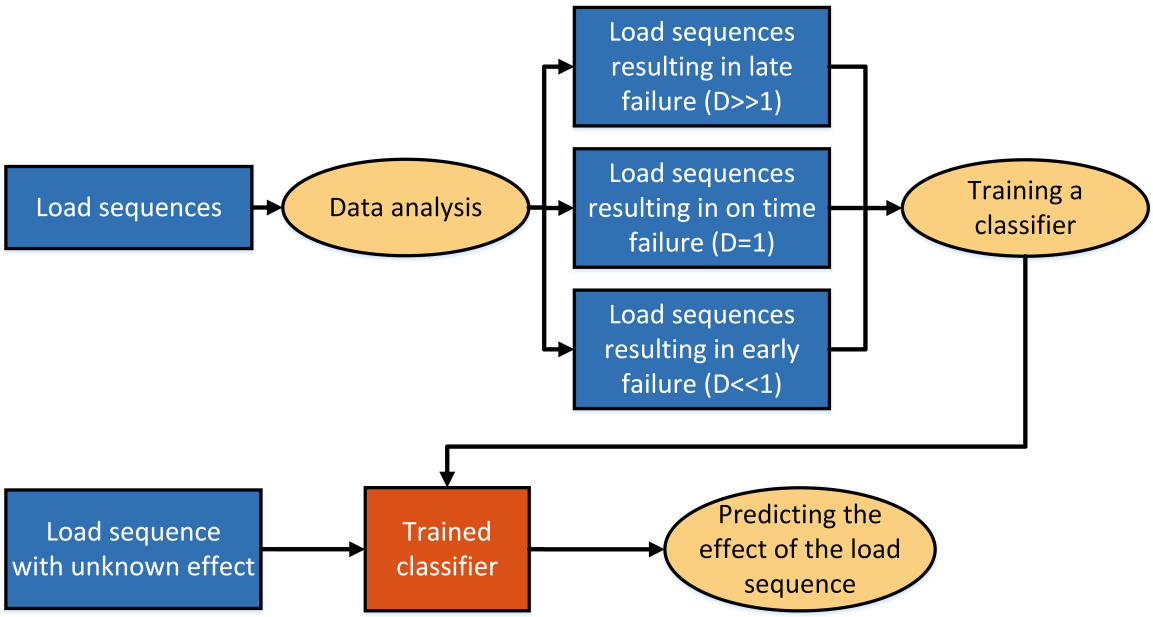

Figure 3.5: Methodology for classifying a load sequence with unknown effect

The SOH is not to be confused with a RUL prediction. SOH is comparable to a cars' notification that, for example, the brakes are 80% worn down and will soon require a maintenance visit to the mechanic.

The SOH prediction is based on similarly damaging load sequences, with a linear interpolation between the starting point of the sequence at 0, and failure at 1. The damage sum is not taken into account in this step. Figure 3.6 shows a schematic diagram of how a SOH prediction is acquired. First, a subset of the original datasets is taken where all load sequences have the same damaging effect.

The end points of those known sequences are mapped to a linear function starting at 0 at the beginning of the sequence and ending at 1 at the end. This function is referred to as the label-function. Now, the load sequences are cut at different endpoints. The label of the shortened sequences is calculated with the label-function. The label of those snippets is the value of the label-function at the end of a sequence. The shortened sequences and their corresponding labels are combined into a data-set. That data-set is used to train a regressor to predict the SOH of a new sequence. The detailed explanation of the involved steps for predicting the SOH follows in chapter 3.5.1.

Figure 3.6 shows a diagram of the proposed structure for the usage of the regressor.

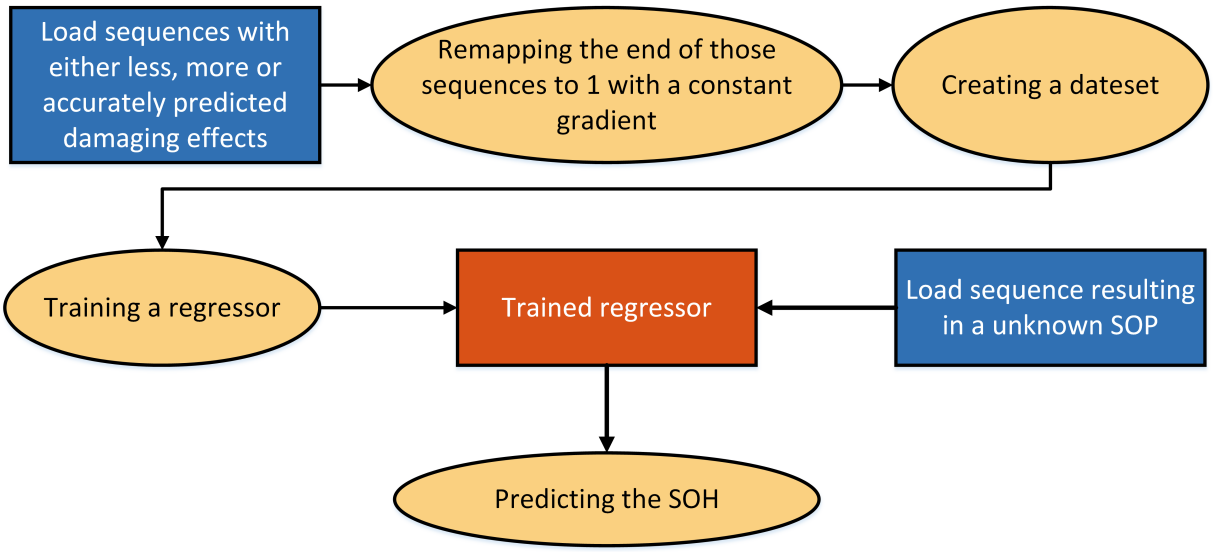

Figure 3.6: Methodology for predicting a SOH based on load sequences with similar damaging effects

In the following, load sequences that have been acquired by STBF are referred to as "finished" load sequences due to the fact that they ended in failure. On the other hand, load sequences that have not yet ended in failure and need to be classified are referred to as "unfinished" load sequences. Only the finished load sequences are used for training the classifier and regressor.

3.4 Preparation of the Classifier

3.4.1 Separation based on Class Label

When it comes to classification, it requires a data-set that is labeled (see Chapter 2.2.4). The recordings provided by the experiments do not contain any clear indicators from which the label can be deduced right away. To acquire the label for each individual sequence, the Basquin-Equation [DIN15] has to be utilized. With the help of this equation, shown in equation 3.1, the maximum load-cycles (N), that an element can withstand at a specific load, can be calculated.

The parameters C, L_a and $-k$ are constants representing the reference value of fatigue strength, the value of the load at one cycle and the slope of the S-N fatigue strength curve [SOL23].

$$N = C * {L_a}^{-k} \tag{3.1}$$

Inputting the result from equation 3.1 in equation 2.1 (from chapter 2.1.3) will return the accumulated damage sum D. Figure 3.7 shows the required steps on how the unlabeled load sequences are analyzed and sorted into categories based on the resulting damage sum D. The class labels and the appropriate range of damage sum D are shown in table 3.1.

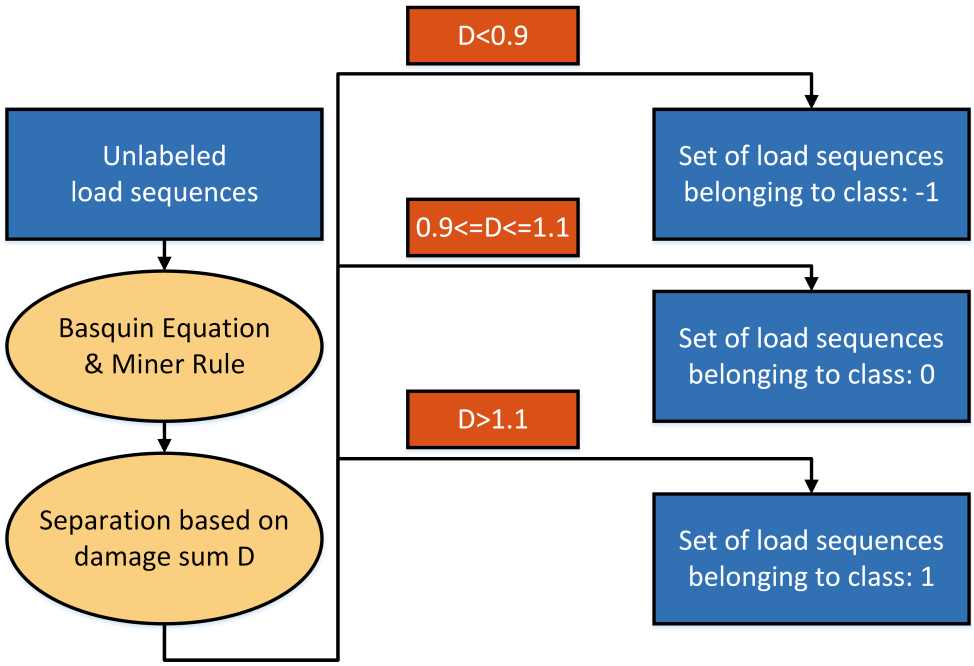

Figure 3.7: Calculation of damage sum D and separation into classes

Damage sum D	Class
D < 0.9	Class: -1
0.9 <= D <= 1.1	Class: 0

Table 3.1: Label assignment based damage sum D at failure

Figure 3.8 gives a more visual representation of how the different load sequences are separated into classes. Note that short sequences can have a higher damage sum D as it is not dependent on just the number of cycles but also on the load level. The end of a sequence is the point where the gear failed.

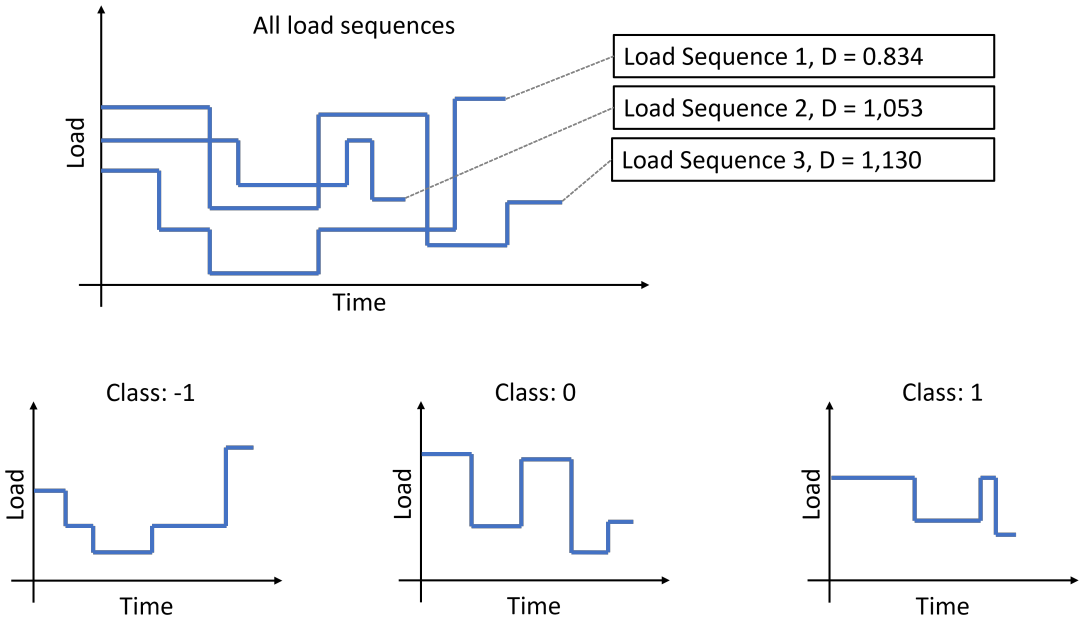

Figure 3.8: Separation of load sequences based on classes

3.4.2 Pre-processing for Classification

Before the classifier can be trained, further pre-processing steps need to be performed. The first is data augmentation (DA), and the second is dimensionality reduction.

The goal of DA, as mentioned in chapter 2.2.3, is to increase the diversity and size of a data-set. This gives the classifier more data to train on and reduces the risk of over-fitting. This step is especially important if the sample size is small. Due to the physical nature of the problem, multiple aspects have to be considered when performing DA.

If the data-set is very limited, DA will only slightly boost the performance, as the inherent problem is the lack of data. Further, if DA is performed, the accumulated damage sum D of the total load sequence must not change significantly. If a load sequence is changed significantly and in such a way alters the damage sum D, the inherit properties of the load sequence can be lost. The classifier relies on characteristics that correlate to the label of a load sequence. Significantly changed characteristics will confuse the classifier in training and lead to poor performance.

The same principle applies to the step of dimensionality reduction. ML models perform better with input vectors that are short and have values in a similar range. One possible option for dimensionality reduction is selecting every n^{th} point in a sequence. When analyzing, if the reduced sequence has the same damage sum D as the original, the numerator of the Miner rule must be multiplied by the step-size that was used to select every n^{th} point. For example, if every 200^{th} point is selected, the step-size is 200. When calculating the damage sum D on the reduced sequence with equation 2.1 the numerator must be multiplied by 200.

Figure 3.9 shows the steps of DA and dimensionality reduction, including a feedback loop to ensure that the augmentation and dimension reduction do not change the damage sum D. It is

important that the comparison is performed after each individual step. In a worst-case scenario, the augmentation and dimensionality reduction change the damage sum in opposite directions. In this case, the final damage sum is equal to the original load sequence, but the characteristics are all lost.

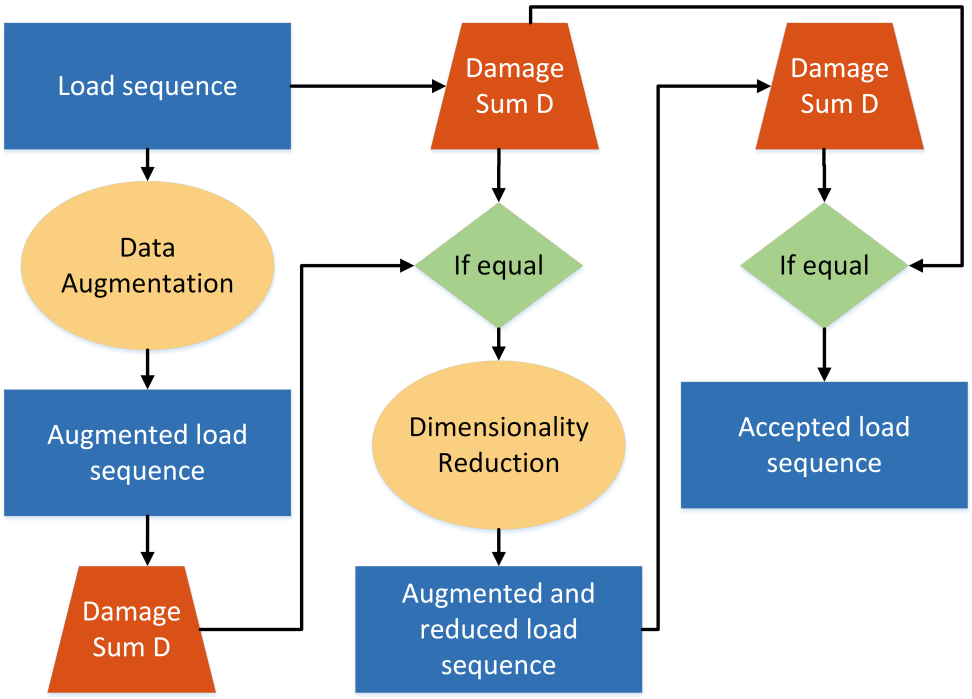

Figure 3.9: Data Augmentation and Dimensionality Reduction with feedback loops

Note that the equality of th damage sum D after DA is the optimal case. Finding DA techniques that vary the load sequences but keep the damage sum D the same is very difficult. To work around this problem, a load sequence is regarded as acceptable if the damage sum D differs by 10 % in any direction.

Figure 3.10 shows the issue of just selecting every n^{th} point for dimensionality reduction. By missing significant portions of the load sequence, peaks and low spots can be missed or accentuated. Multiplying the number of cycles in the Miner rule with n, is the same as expanding every cycle by n and using the Miner rule without multiplication. The two graphs in 3.10 show that the expanded load sequence has significant differences and that all characteristics have been lost. By comparing the damage sum at each step to the original, such cases can be found and not taken into the training data-set. Another way of reducing this problem is to select a smaller step-size which in turn will increase the feature vector size that will be used as an input.

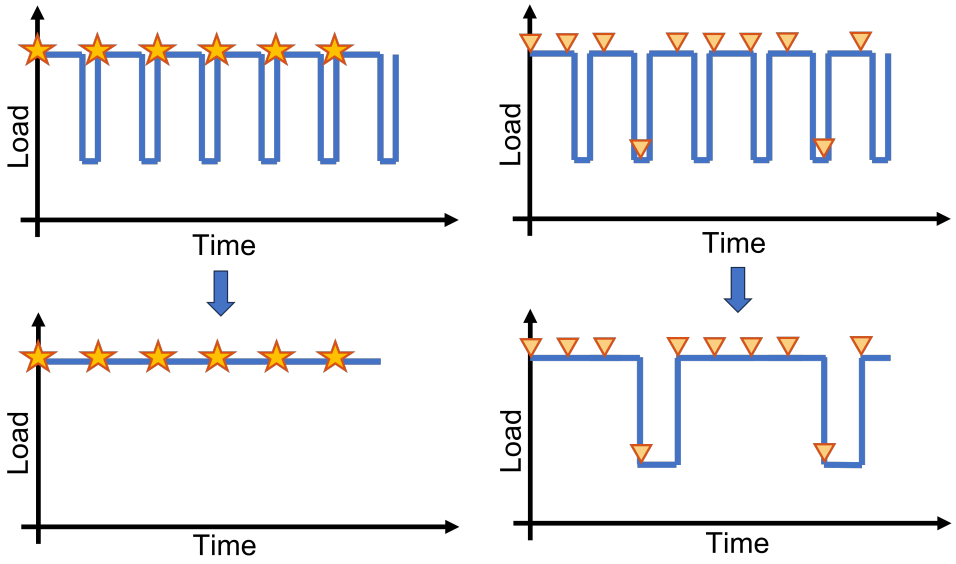

Figure 3.10: Danger of Dimensionality Reduction by selecting equidistant points

3.4.3 Usage of the Classifier

The classifier is used as a method to classify an unfinished load sequence, whether it is part of the more damaging, less damaging ones or correctly assessed ones. After the performed steps of separation into classes, DA and dimensionality reduction, the load sequences require a last pre-processing step. At the moment, the load sequences are in their full length and have a different damage sum D. If an unfinished sequence, of a gear that has not yet suffered fatigue damage needs to be classified, the comparison has to be made with sequences that have the same damage sum D as the unfinished sequence. In other words, if a current sequence is at damage sum D = 0.6 and it is of interest if this is a correct estimate, it needs to be compared to other load sequences at D = 0.6.

To do so, all sequences are cut (shortened) so that they all end at the same damage sum D. With those sequences and their labels, the classifier is trained. After training, the classifier can take the unfinished sequence that underwent the same dimensionality reduction as the rest and predict a class-label. Based on the predicted class label, a conclusion can be drawn about whether the calculated damage sum D based on the Miner rule is more on the conservative or lenient side. Figure 3.11 shows the steps of the last pre-processing steps and the classification of a sequence.

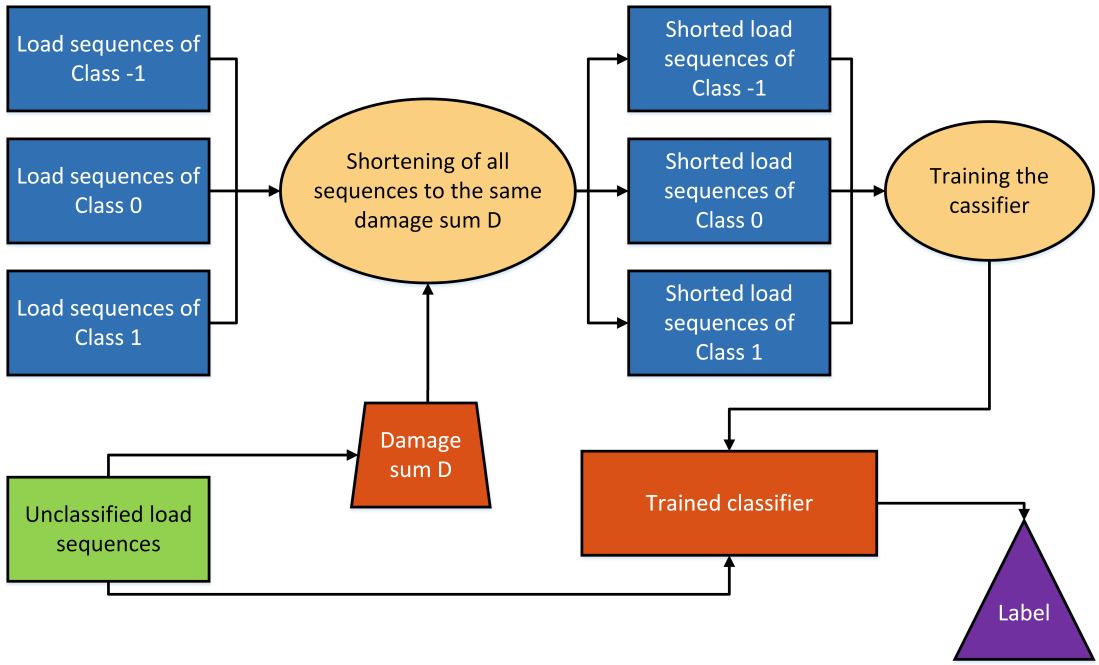

Figure 3.11: Load sequence adaption, training and final classification

Figure 3.12 gives a visual representation of how load sequences of class 1 (D>1.1) are adapted when the unfinished sequence has the damage sum D of 0.6 according to the Miner rule. All recorded points after D=0.6 are deleted in every load sequence. By doing so, the sequences are shortened. These sequences are then used as training data to predict the label of the unlabeled sequence. The same is done to the sequences of classes 0 and 1.

Note that if an unlabeled sequence is past a certain damage sum D, the classification will become more uncertain. For example, if a sequence has a damage sum D = 0.85 but most of the sequences in class -1 end at 0.8, there are significantly fewer data samples that can be used for training in that class.

For a classification of a load sequence that has a very low damage sum D (for example 0.3) the classification also becomes very uncertain, as it is possible that not enough characteristics are present in a load sequence that could place it in one category with high certainty. So there exists a sweet spot that is determined by the number of training-elements in each class and the damage sum D of the unknown label.

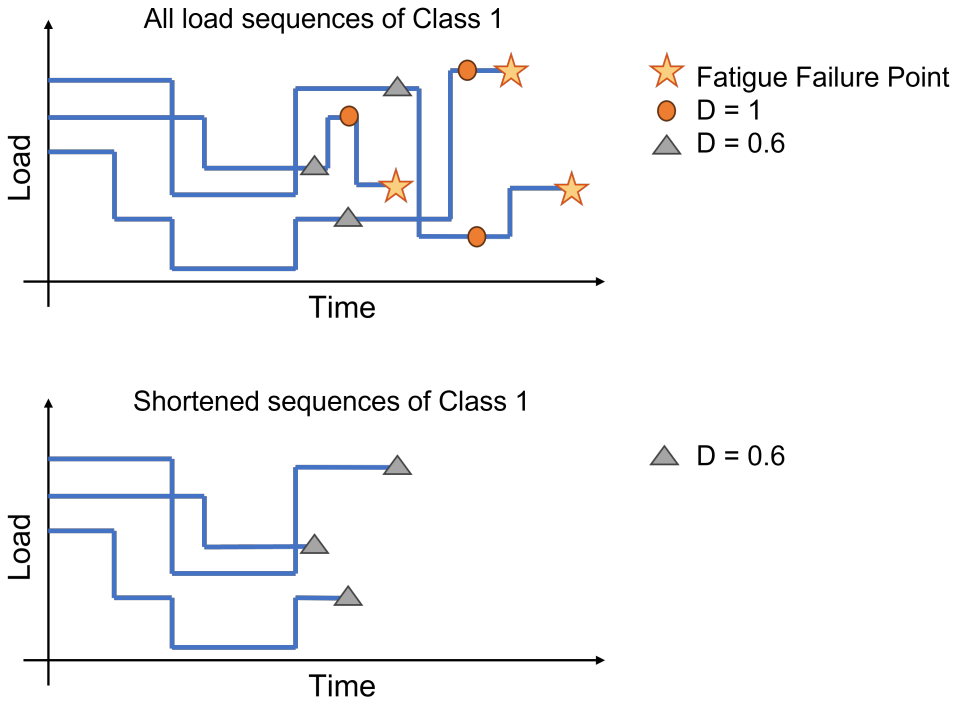

Figure 3.12: Load sequence shortening

3.4.4 Problems in Classification of Sequences

One of the problems in classifying sequences is the difference in length of those sequences. The original recordings can have a different number of cycles, which makes it difficult to use them as input-vectors. Most algorithms require a constant input-vector size for training and validation. This problem can be circumvented by pre-filling or appending each augmented and reduced sequence with zeros to a uniform length, or using an approach that can work with sequences of different sizes.

The only problem remaining is the prediction of an unseen sequence that might be longer than any other sequence seen in training. This problem can be solved by pre-pending more zeros to create a buffer for the ability to classify longer sequences than those present in the training data.

3.5 Regressor for SOH-Prediction

The regressor is used as a linear extrapolation model that calculates the SOH of a gear. The SOH can be interpreted as a value for how soon the gear is expected to fail. For example, if the Miner rule calculates a damage sum D equal to 0.8, the classifier assigns a class label of 1 (not damaging load history), and the regressor calculates a SOH of 0.9, the gear is expected to be able to withstand higher loads but not for a very long time. In that case, it is expected that the damage sum D will accumulate past unity, but will be limited in the number of further cycles it can withstand.

3.5.1 Data-set for Regressor

If an unfinished load sequence is classified into a class, for example, class 1, the regressor will only be trained on the load sequences belonging to that class. Each augmented load sequence is iteratively selected, and n different, randomly determined points are chosen. These points are transferred to the label function of that sequence to determine the label (SOH) of that sequence snipped from the beginning up to that point. The selected snippets are then saved and used as training data for the regressor. Figure 3.13 shows the process of creating data samples from one load sequence.

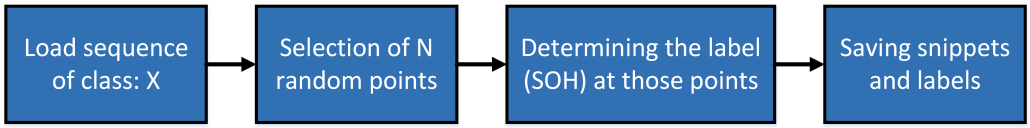

Figure 3.13: Data-set generation process for the regression data-set

Figure 3.14 gives a visual example of how three elements and the appropriate labels are created from one load sequence. The number of created elements can be freely defined.

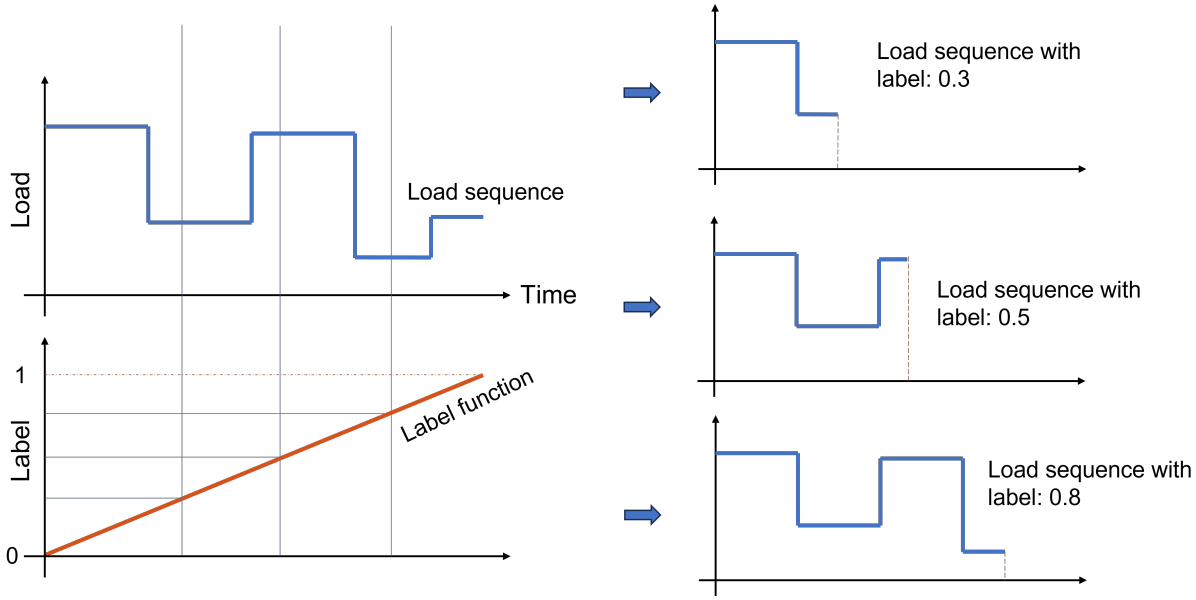

Figure 3.14: Creation of three elements with the help of the label function

3.5.2 Training the Regressor

The created training data-set has the same structure as the classification trainig-set. The sections are now of different lengths, not only because of the different number of cycles but also because of the different selected points. The regressor must either be able to accept input vectors of different sizes or each sequence needs to be pre-filled or appended with zeros (zero-padding) to bring all elements to the same length.

3.6 Summary of the Proposed Method

The goal of the proposed method is to get a confidence value regarding the calculated damage sum D according to the Miner rule. It achieves that by first classifying a sequence into a category that represents the influence of the order of the loads. By doing so, it can be determined if the Miner rule is giving a conservative or lenient estimate. The classification is based on a supervised learning algorithm that uses load sequences that failed before and after reaching an accumulated damage sum D equal to unity. The second step is a regressor that predicts a state-of-health, which is a value calculated by extrapolating the expected failure point in time based on the load sequences that have the same effect based on the order of loads. With these two values, the confidence in the calculated damage sum D can be evaluated. Table 3.2 gives an overview on how to interpret the different combinations of the two values.

Class	SOH	Effect
-1 (Very damaging)	0,5	will fail at D<1, but can withstand many cycles with low loads
1 (less damaging)	0,5	will fail at D>1, and can withstand many cycles with high loads
-1 (Very damaging)	0.9	Will fail very soon
1 (less damaging)	0.9	Will fail very soon but exceed D=1 if future loads are high

Table 3.2: Effect of the predicted class and SOH

4 Validation

This chapter covers the implementation and results of the proposed method in chapter 3.

4.1 Implementation

4.1.1 General Information regarding the Implementation

The method is implemented in Python 3.9.9 for its widespread availability of libraries for ML-algorithms. The libraries from which the classifiers and regressors are taken are:
"scikit-learn" [sci23], "TSlearn" [tsl23] and "xgboost" [XGB23]. Multiple models from those libraries are tested and compared regarding their performance.

4.1.2 Data Cleanup

The first step is the cleanup of the data and its translation into a more flexible format. The recordings of the load sequences are provided in a text-file format that includes information such as displacements of the piston, cycle-time and cycle-count. Figure 4.1 shows all the labels of the recorded values in the txt-file and their interpretation. All information is disregarded except the net force at each cycle-peak. The net force is the absolute clamping force (lowest acting force) plus the recorded absolute peak (highest acting force). The order of loads is kept, and the absolute values are transformed into an array that is stored as a NumPy file (np-file). 14 load sequences are available, which are transformed into 14 np-files and stored in a new folder. The np-file format allows for faster read times and takes up less space on the hard drive than the original txt-files. The names are not important and are thus also replaced by a simple index.

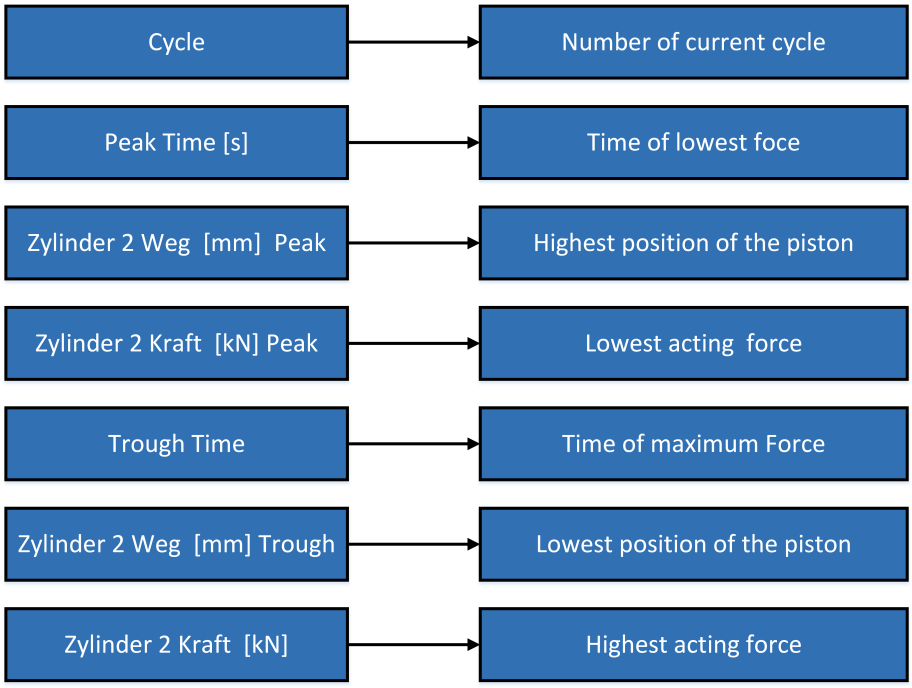

Figure 4.1: Recorded parameters in STBF

4.1.3 Data Augmentation

The data augmentation process starts with calculating the damage sum D with the help of the Basquin equation (see equation 3.1) and the Miner rule (see equation 2.1). The parameters for the Basquin equation were provided and shown in table 4.2. Only forces higher than the endurance limit are selected when calculating the accumulated damage. The calculated damage sum D according to the Miner rule for each available file are shown in table 4.1

Load Sequence Nr.	Damage sum D
0	2.16
1	1.18
2	1.29
3	0.92
4	3.4
5	1.18
6	1.4
7	1.01
8	1.06
9	0.75
10	0.68
11	1.22
12	0.73
13	1.1

Table 4.1: Damage sum D at failure of available load sequences

Parameter D	Value
Slope of the S-N fatigue strength curve = k	6.33048
Reference value of fatigue strength = C	11670367740000000
Value of the load at one cycle = L	values from array
Endurance limit	56.618 kN

Table 4.2: Parameters of the Basquin-equation and Endurance Limit

The first step in DA is the addition of random noise. A np-file is loaded and the damage sum D is calculated. That value serves as a reference to compare if the augmentation significantly changes the damage sum D. The random noise is in the form of a vector that has the same length as the loaded load sequence. The individual noise values follow a Gaussian distribution with a mean of 0 and a standard deviation of 0.2. The values of the noise vector are also in kN and thus can be simply added. Figure 4.2 shows a load sequence in its unedited form without the addition of noise. Figure 4.3 shows the same load sequence with the additional noise vector. As can be seen, the damage sum D is not changed enough to be noticeable when rounding to two decimal places, even though there is a clearly visible difference in the load sequences.

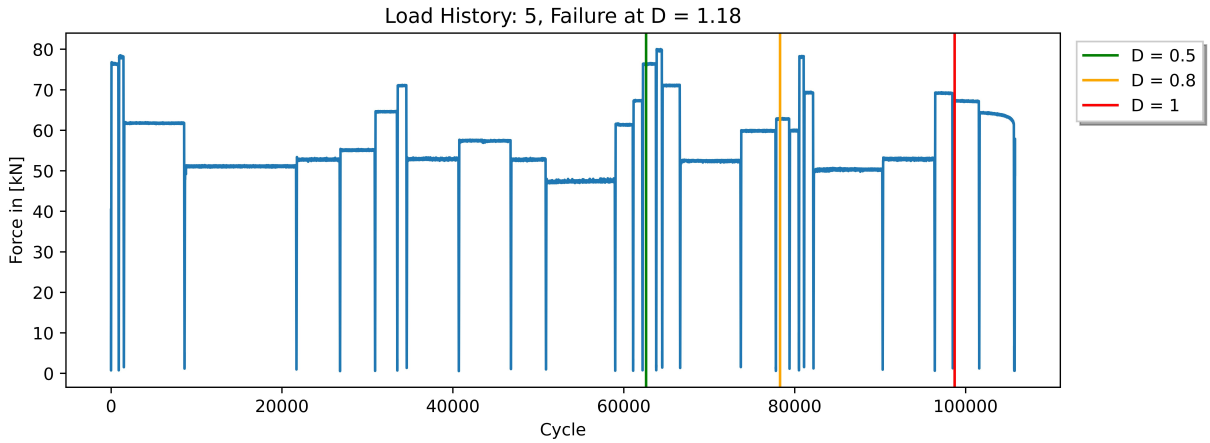

Figure 4.2: Load sequence without DA

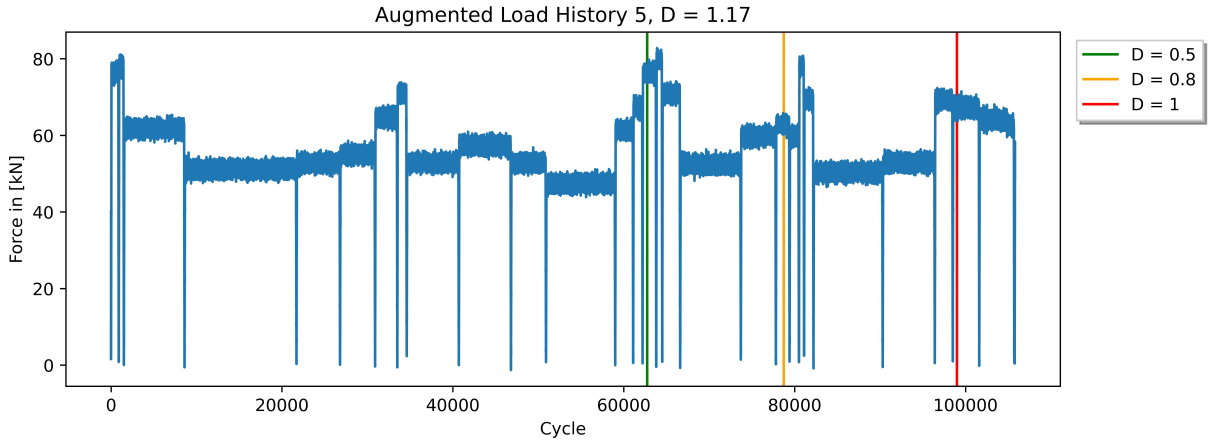

Figure 4.3: Load sequence with additional noise

The second performed step is a global shift of the whole loading sequence up or down by up to 500 N. This is done by sampling a uniform distribution between -1 and 1 and multiplying by 0.5. This constant is added to every array element. Figure 4.4 shows the shifted load sequence with the previously added noise. In this case, the damage sum D is altered either in the negative or positive direction. Again, no significant changes occurred regarding damage sum D.

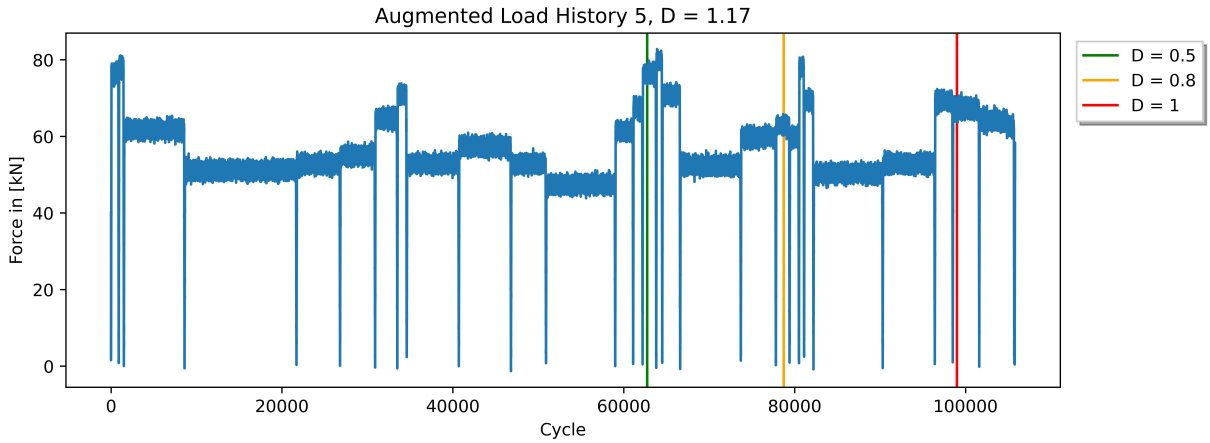

Figure 4.4: Load sequence with noise and shift

The third step is a section-wise random shift. A random section of the array with a length of 1 % of the total length is selected. This section is multiplied by a random factor taken from a uniform distribution between 0.5 and 1.5. So the force in that section is either increased or decreased by up to 50 %. This step is performed three times on one load sequence. This step effectively changes the values of 3 % of the sequence.

In this step, the damage sum D can change significantly. The most significant changes occur when the selected areas are overlapping and scaled in the same direction. In a worst-case scenario, the selected section includes a proportionally high load and is amplified it by a factor of up to 3.375 (1.5^3). This scenario is highly unlikely and in the case of a significant alteration of the damage sum D the augmented load sequence is discarded as explained previously in figure 3.9.

Figure 4.5 shows how the load sequence is changed by applying that DA step. Notice the down-shifted area around the 92,000th cycle.

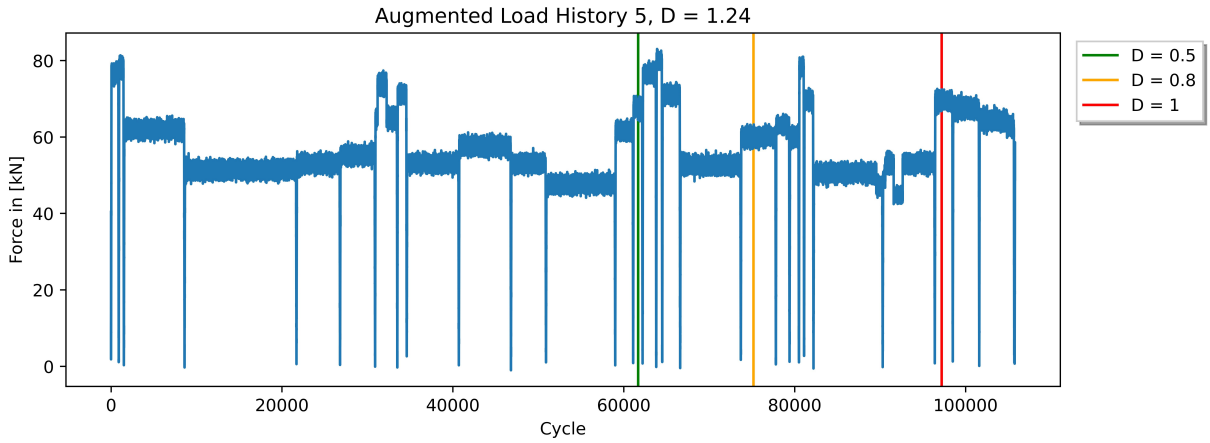

Figure 4.5: Load sequence with partial shift

The fourth step is a partial cutout, where a random section of length of 1 % of the total length is cut out of the sequence. In this step, the damage sum D is always decreased. Figure 4.6 shows how the loading sequence is transformed with partial cutouts.

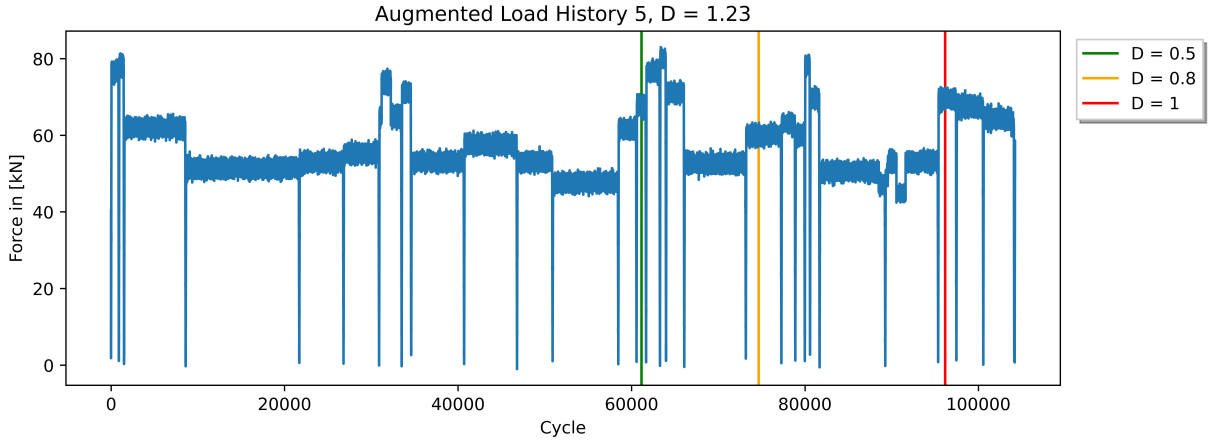

Figure 4.6: Load sequence with partial cutouts

The next step is the addition of a vector whose values are either going from positive to negative or from negative to positive over its length. It is essentially weighing the values of the load sequence over its length. By adding that vector to the load sequence, some values are increased while others are decreased.

At first, a random variable decides if the sequence is tilted up (from negative to positive) or down (going from positive to negative). In the case of an upward shift, the start values of that vector are random variables selected from a uniform distribution in the range [-0.2, 0]. The last value is selected from a uniform distribution in the range [0, 0.2]. In the case of a downward shift, the ranges are switched. The values between are interpolated accordingly with a constant step size.

Figure 4.7 shows all the previously mentioned DA-steps combined.

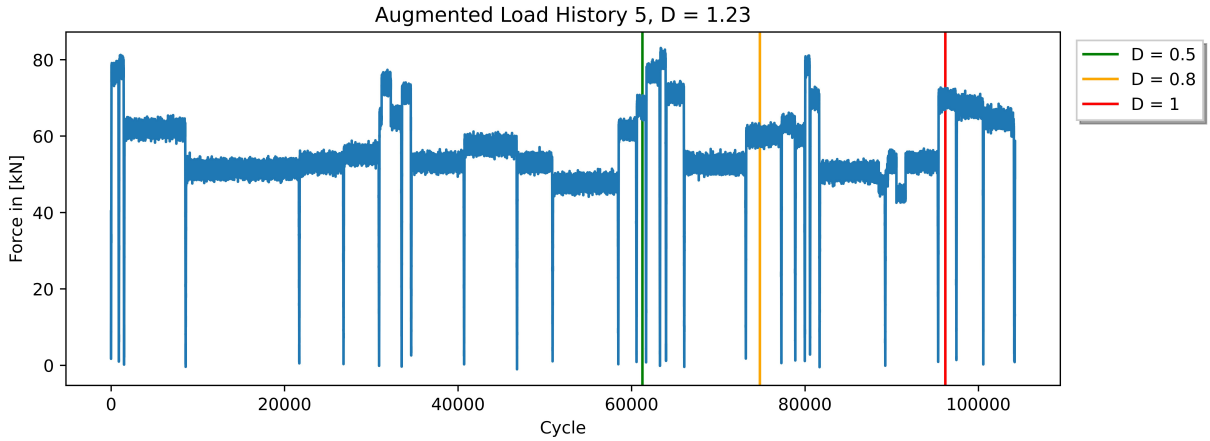

Figure 4.7: Load sequence with tilt

When comparing the original and the augmented load sequence, the visual difference is low. This is due to the fact that damage sum D must not change more than 10 % of its original value. Only load sequences that fulfill this criteria are accepted into the training datasets. To filter out all sequences that are off by 10 % the original damage sum D and damage sum D after augmentation are compared. If the ratio falls within [0.9, 1.1] the sequence is accepted and the final step is applied.

The last step is dimensionality reduction, as described in chapter 3.4.2. In this case, every 300^{th} point in that sequence is kept. This step is effectively reducing the length of the sequence by a factor of 300. For the last comparison of the damage sum D, the denominator in the Miner rule must be multiplied by the step size. Again only if the ration falls within [0.9, 1.1], the sequence is accepted into the data-set.

Figure 4.8 shows the load sequence in its reduced form. Notice that the characteristics of the original load sequence are still present.

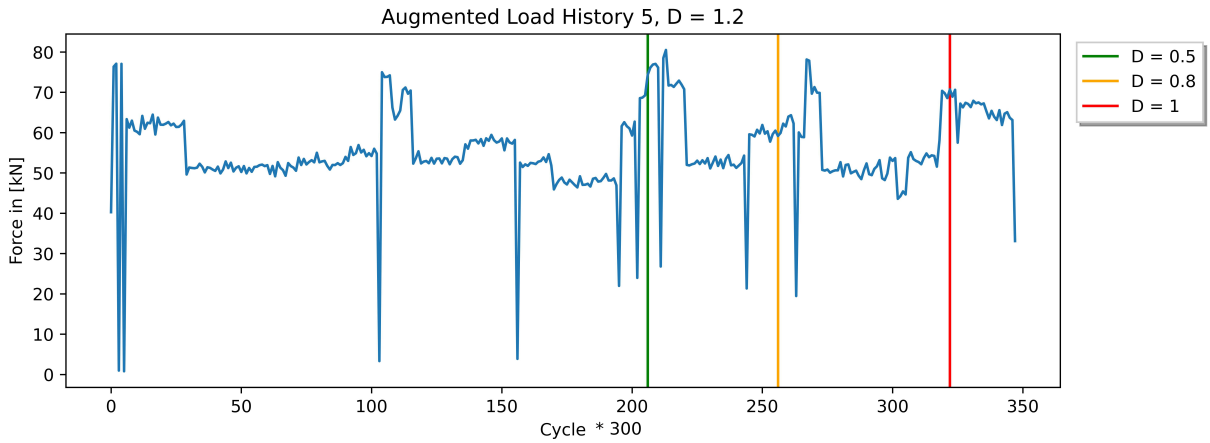

Figure 4.8: Load sequence after dimensionality reduction

Each load sequence is augmented with the described procedure 25 times. In the end, 25 individual instances can be present in the datasets if none of the augmented ones are filtered out. The accepted load sequences are saved as a np-array in a new separate folder. The name of the np-file follows the following pattern: [class_idx.np]. "class" is the class to which the loading sequence belongs to, according to table 3.1 and "idx" is a counting index that is increasing from 0 to n to keep track of the number of data samples.

Figure 4.9 shows only three out of those 25 augmented sequences in one plot to highlight the differences in each sequence. Notice how the points of damage sum D of 0.5, 0.8 and 1 vary in their positions, but the characteristics of the original load sequence are kept.

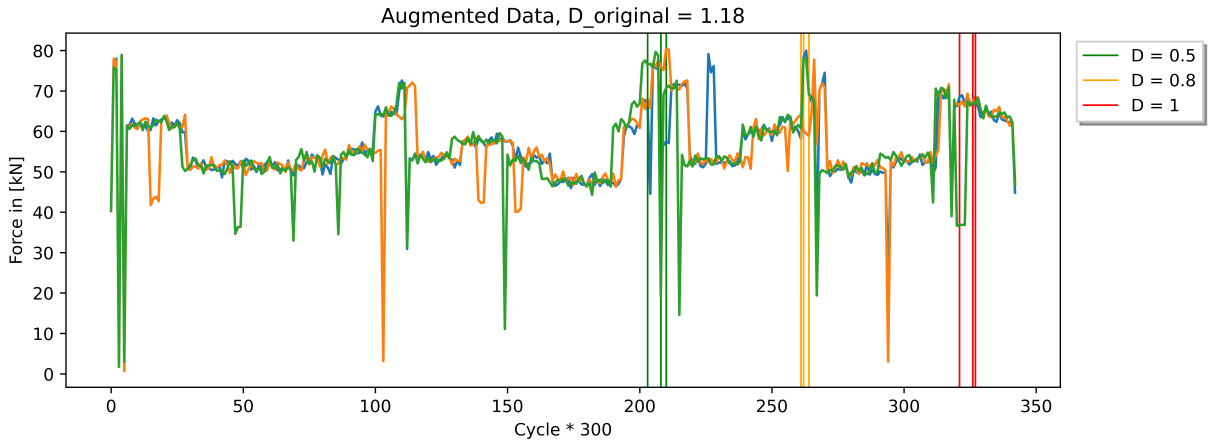

Figure 4.9: Differences in augmented load sequences

Figure 4.10 shows all performed DA-steps in a schematic diagram.

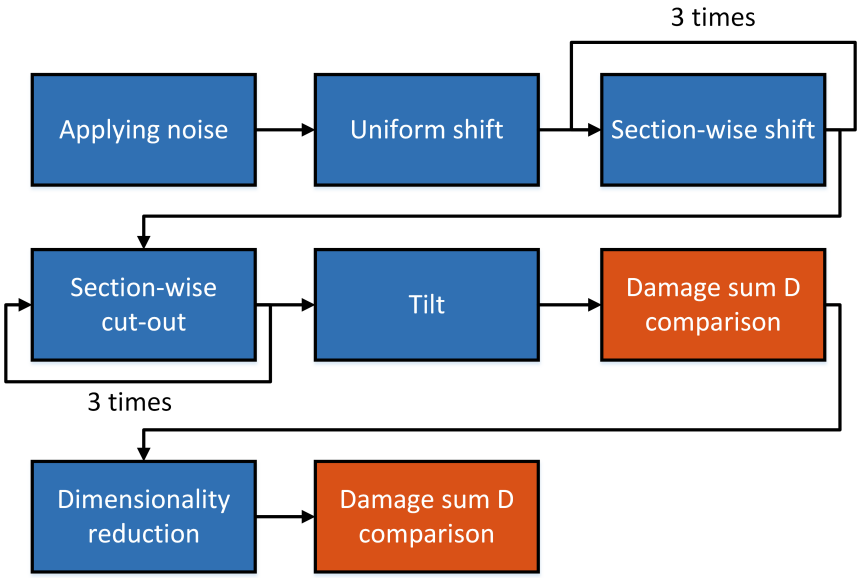

Figure 4.10: Augmentation procedure

From the original 14 load sequences, on average, 200 augmented sequences are created. This means that 150 out of 350 load sequences do not meet the requirement that the damage sum D is not significantly different from the original load sequence. The goal is to augment the load sequence as much as possible while at the same time retaining the damage sum D and the characteristics of every load sequence.

The average length of the load sequences after dimensionality reduction is 320 data points. The shortest load sequence has 66 data points, and the longest one has 1000 data points. This large difference comes from the fact that load sequence number 4 (see table 4.1) failed at a very high damage sum D while load sequence number 12 failed at a very low damage sum D.

One challenge when applying the selected DA-techniques is that a small alteration in damage sum D in load sequences can have a very large relative change compared to the original damage sum D. For example, sequence number 12 has a damage sum D of 0.73 and after DA a damage sum of 0.82. The absolute change in damage sum D is 0.09. But when looking at the ratio, (0.82/0.73 = 1.12) the augmented load sequence is not accepted into the training set. Thus, almost no augmented elements of sequence 12 are present in the training data set. The same problem is present, but less pronounced, with other sequences with low damage sum D.

4.2 Training and Validation

When having an unfinished load sequence that needs to be analyzed, chapters 3.4.2 and 3.5.1 explain that the classifier and regressor are trained specifically for that individual load sequence. This is because the training set is adapted to the current damage sum D of the unfinished load sequence.

For testing purposes, the damage sum D (which would have come from the unfinished load sequence) is manually set to 0.6 The success of the classifier and regressor is analyzed by splitting the data into a test-set and training-set. 20 % of the data is given to the test-set and 80 % to the training-set.

The training-set contains on average: 50 elements of class -1, 30 elements of class 0 and 90 elements of class 0. This ratio comes directly from the distribution of the original data-set where most load sequences failed at a damage sum D higher than 1.1.

4.2.1 Training of the Classifier

To train the classifier, the augmented load sequences are loaded into one list of arrays. The label of the array is extracted from the names of the files created in the augmentation step and also stored in an array.

The load sequences are in their full length and need to be shortened to the length where the damage sum D is equal to 0.6 to be comparable to the unfinished load sequence. Note that in this case, the unfinished load sequence corresponds to the sequences in the test-set.

The dataset is divided into a training- and test-set with the help of a prebuilt train-test-split function taken from [sci23]. Table 4.3 shows the ten classifiers that are implemented to compare their individual performance:

1) KNeighborsTimeSeriesClassifier [tsl23]	2) TimeSeriesSVC [tsl23]
3) AdaBoostClassifier [sci23]	4) GradientBoostingClassifier [sci23]
5) RandomForestClassifier [sci23]	6) XGBoost[XGB23]
7) Neural Net [sci23]	8) GaussianNB [sci23]
9) QuadraticDiscriminantAnalysis [sci23]	10) KNeighborsClassifier [sci23]

Table 4.3: Implemented Methods for Classification

Only the "KNeighborsTimeSeriesClassifier" and "TimeSeriesSVC" are capable of accepting input vectors that have a varying length. To use the remaining classifiers, the load sequences are brought to the same length by pre-pending zeros, so that the overall length is equal to the longest sequence. Note that the length is defined by the longest load sequence in both of the datasets (train- and test-set).

4.2.2 Training of the Regressor

The training set of the regressor is dependent on the result of the classifier. For analysis purposes, the performance is analyzed by predicting the SOH on all three possible classes.

For a predicted class, for example class 1, only the load sequences are loaded that belong to that class. In each load sequence, a random point is selected, and the label is calculated by the label function as described in chapter 3.5.1. The array is shortened to the randomly selected point and stored in a list. The corresponding label is stored in an array. From each load sequence, four such frames are selected.

The analyzed regressors for predicting the SOH are:

- AdaBoostRegressor [tsl23]
- RandomForestRegressor [tsl23]

The library "TSlearn" [tsl23] contains a regressor called "TimeSeriesSVR", which is capable of working with input vectors of different lengths. The issue is that the maximum length must not exceed 450 elements. Because some arrays exceed this length, only two selected regressors from SKlearn [sci23] are tested. To bring all arrays to the same length, a zero-padding method is employed, just as in the classification approach.

4.3 Analysis and Discussion of the Results

4.3.1 Results of Classification

To get an accurate evaluation of the performance of the selected classifiers, each model is trained and tested 25 times on a different data-set. This means that the data augmentation step of all load sequences is performed every time before the training of a classifier starts. By doing that, the train- and test-set are varied. Another way to get the average performance is k-fold cross-validation as described in chapter 2.2.3. Due to the ability to generate variations in data, the k-fold cross-validation approach is disregarded. By averaging the performance over 25 iterations on the test-set, an accurate average performance is calculated.

The average individual performance of all tested classifiers is shown in figure 4.11. A score of 1 means a correct classification of all sequences. The value in the plot is a ratio of correctly predicted classes to all predictions.

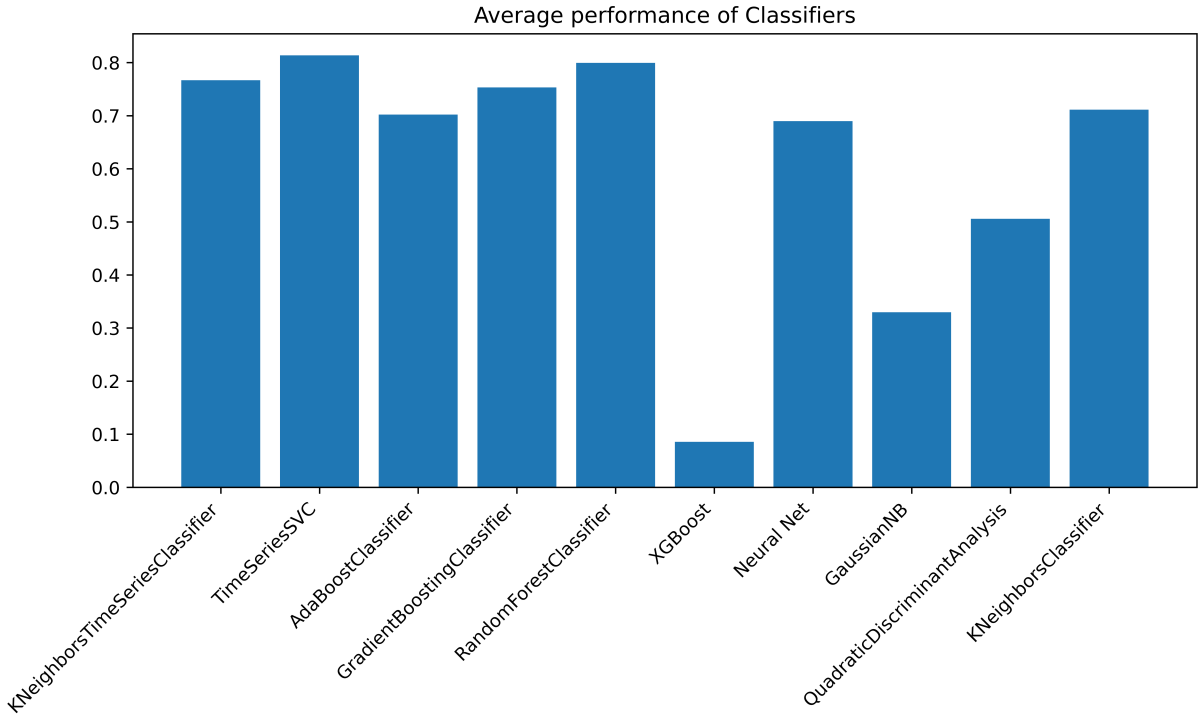

Figure 4.11: Average performance of predicting the correct class for each classifier

When looking at the results, most of the classifiers are capable of predicting the correct class for the load sequences in the test-set of up to 80 % of the time. It is noticeable that the two first classifiers that are working with input vectors of varying length are as capable as the classifiers that require constant dimensions. The best performance is achieved by "TimeSeriesSVC" from [tsl23] which predicted the correct class 81.3 % of the time. The worst performance was achieved by "XGboost" with only 8.5 %. This is a noticeable result because an average score of 33 % is expected when the labels are predicted at random.

Table 4.4 shows the analyzed classifiers sorted by performance.

Classifier	Average performance
TimeSeriesSVC	81.3 %
RandomForestClassifier	79.9 %
KNeighborsTimeSeriesClassifier	76.6 %
GradientBoostingClassifier	75.3 %
KNeighborsClassifier	71.1 %
AdaBoostClassifier	70.1 %
Neural Net	68.9 %
QuadraticDiscriminantAnalysis	50.5 %
GaussianNB	32.9 %
XGBoost	8.5 %

Table 4.4: Average performance of classifiers

4.3.2 Results of Regression

The regressors are trained for each possible class and evaluated on their performance. The performance is averaged over 25 trainings to get an accurate average score for the regressors. The score is obtained by averaging the mean-square error of the prediction for each class.

Figure 4.12 shows the result of the averaged training. The predictions of the correct SOH in classes -1 and 1 are more accurate than in class 0. One possible explanation for this performance is the previously mentioned disparity in number of elements in each class. Class 1 and class -1 have more data points and thus allow for a better prediction. Both regressors show similar performance in each class.

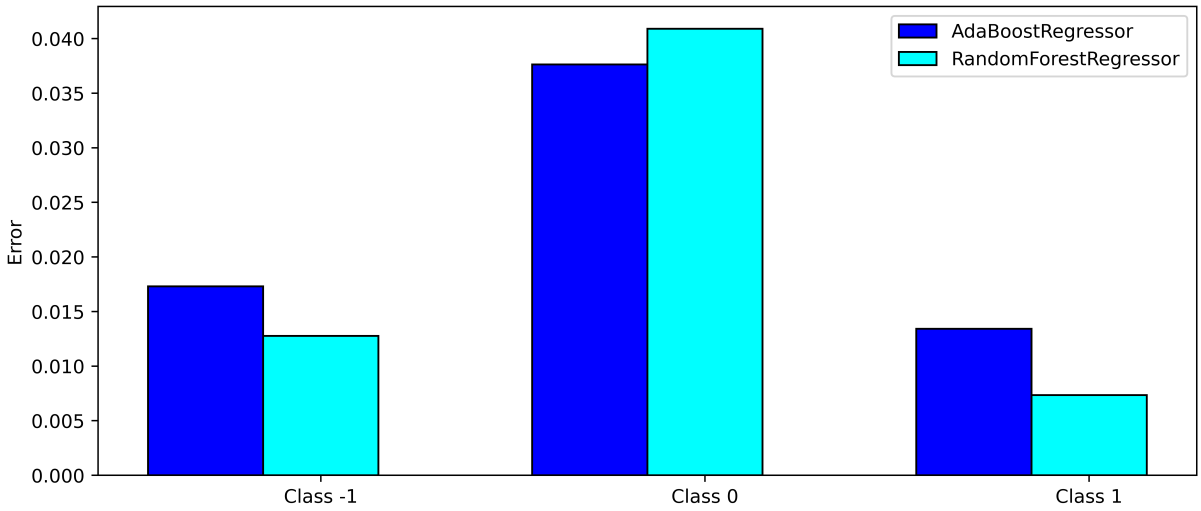

Figure 4.12: Average performance of each regressor for predicting the SOH in each class

4.3.3 Critical Discussion of Results

The tested methods prove to return accurate results in classification and regression. One possible reason for that is that the implemented DA steps do not alter the structure of the loading sequence enough to be seen as a completely different load sequence. Thus, if an augmented element of a load sequence is in the training-set and testing-set, the ML models do not need to have the ability to generalize to predict the correct class for that sequence.

When predicting the SOH the inputs from test- and training-set have significant differences in length. Thus the same problem of over-fitting is not present. For a more accurate result and better performance of the ML models more data is necessary. As mentioned before, only 14 load sequences were available which makes the training of the ML models to generalize well, a challenging task.

5 Conclusion

5.1 Summary of Results

The goal of this thesis is to develop a method that is able to provide a confidence value in the calculated state-of-damage by specifically considering the history (order) of the loads that a machine element is subjected to. This thesis proposes a method that can classify a load sequence into one of three possible classes, depending on the effect of the load history. Further, a second indicator called state-of-health is added to analyze how much a machine element is worn down. With the combination of these two values, the state-of-damage is now supplied with more information and can be viewed from a different perspective. The analyzed classification methods return very promising results and are capable of correctly classifying a load sequence in 80 % of cases. The selected regression models are also capable of predicting the state-of-health with very low error margins. In conclusion, this method provides a very capable and robust alternative to nonlinear damage accumulation models that provides accurate results and is easily applicable to every machine element.

5.2 Outlook

One of the most important aspects when it comes to ML approaches is the amount and quality of the data. To get an even more accurate result of the predicted classes and state-of-health the supply of more data is the most important step. Further, it is possible to add more DA methods to increase the model's ability to generalize. Another possible area for further research is the prediction of the SOH not with a simple linear interpolation but with different functions depending on the output of the previous classification.

References

Standards

[DIN15] DIN 50100 Schwingfestigkeitsversuch-Aktueller Stand der Überarbeitung. (2015).

[ISO19] ISO 6336-6:2019(E): Calculation of load capacity of spur and helical gears: Part 6: Calculation of service life under variable load. (2019).

Papers, Books, Dissertations, Thesises

[Ada20] Adasooriya, N. D. ; Pavlou, D. ; Hemmingsen, T.: Fatigue strength degradation of corroded structural details: A formula for S–N curve. In: *Fatigue & Fracture of Engineering Materials & Structures* 43 (2020) Nr. 4, pp. 721–733. ISSN: 8756-758X. DOI: 10.1111/ffe.13156.

[Ahs21] Ahsan, M. ; Mahmud, M. ; Saha, P. ; Gupta, K. ; Siddique, Z.: Effect of Data Scaling Methods on Machine Learning Algorithms and Model Performance. In: *Technologies* 9 (2021) Nr. 3, p. 52. ISSN: 2227-7080. DOI: 10.3390/technologies9030052. URL: https://www.mdpi.com/2227-7080/9/3/52.

[Alz21] Alzubaidi, L. ; Zhang, J. ; Humaidi, A. J. ; Al-Dujaili, A. ; Duan, Y. ; Al-Shamma, O. ; Santamaría, J. ; Fadhel, M. A. ; Al-Amidie, M. ; Farhan, L.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. In: *Journal of big data* 8 (2021) Nr. 1, p. 53. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8.

[Ami18] Amiri, Roohollah and Mehrpouyan, Hani and Fridman, Lex and Mallik, Ranjan and Nallanathan, Arumugam and Matolak, David: A Machine Learning Approach for Power Allocation in HetNets Considering QoS. In: (2018).

[Asa20] Asadi, B. ; Jiang, H.: On Approximation Capabilities of ReLU Activation and Softmax Output Layer in Neural Networks. (2020). URL: http://arxiv.org/pdf/2002.04060v1.

[Bal19] Baldominos, A. ; Saez, Y. ; Isasi, P.: A Survey of Handwritten Character Recognition with MNIST and EMNIST. In: *Applied Sciences* 9 (2019) Nr. 15, p. 3169. DOI: 10.3390/app9153169.

[Ban21] Bandara, K. ; Hewamalage, H. ; Liu, Y.-H. ; Kang, Y. ; Bergmeir, C.: Improving the accuracy of global forecasting models using time series data augmentation. In: *Pattern Recognition* 120 (2021), p. 108148. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2021.108148. URL: https://www.sciencedirect.com/science/article/pii/s0031320321003356.

[Bap17] Baptista, C. ; Reis, A. ; Nussbaumer, A.: Probabilistic S-N curves for constant and variable amplitude. In: *International Journal of Fatigue* 101 (2017), pp. 312–327. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2017.01.022. URL: https://www.sciencedirect.com/science/article/pii/s0142112317300300.

[Bat19] Batta Mahesh: Machine Learning Algorithms -A Review. (2019). DOI: 10.21275/ART20203995. URL: https://www.researchgate.net/profile/batta-mahesh/publication/344717762_machine_learning_algorithms_-a_review.

[Bel22] Bellows, R. S. ; Bain, K. R. ; Sheldon, J. W.: Effect of Step Testing and Notches on the Endurance Limit of Ti-6Al-4V. In: *ASME 1998 International Mechanical Engineering Congress and Exposition* (2022), pp. 27–32. DOI: 10.1115/IMECE1998-1138.

[Ben02] Benedetti, M.: Influence of shot peening on bending tooth fatigue limit of case hardened gears. In: *International Journal of Fatigue* 24 (2002) Nr. 11, pp. 1127–1136. ISSN: 0142-1123. DOI: 10.1016/S0142-1123(02)00034-8. URL: https://www.sciencedirect.com/science/article/pii/s0142112302000348.

[Ber13] Bergstra, J. ; Yamins, D. ; Cox, D.: Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms. (2013). URL: https://pdfs.semanticscholar.org/d4f4/9717c9adb46137f49606ebbdf17e3598b5a5.pdf.

[Ber21] Bertolini, M. ; Mezzogori, D. ; Neroni, M. ; Zammori, F.: Machine Learning for industrial applications: A comprehensive literature review. In: *Expert Systems with Applications* 175 (2021), p. 114820. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.114820. URL: https://www.sciencedirect.com/science/article/pii/s095741742100261x.

[Bis06] Bishop, C. M.: Pattern recognition and machine learning. Computer science. New York, NY: Springer, (2006). ISBN: 978-0387-31073-2.

[Bur18] Burhan, I. ; Kim, H.: S-N Curve Models for Composite Materials Characterisation: An Evaluative Review. In: *Journal of Composites Science* 2 (2018) Nr. 3, p. 38. ISSN: 2504-477X. DOI: 10.3390/jcs2030038. URL: https://www.mdpi.com/311166.

[Cai21] Cai, S. ; Wang, Z. ; Wang, S. ; Perdikaris, P. ; Karniadakis, G. E.: Physics-Informed Neural Networks for Heat Transfer Problems. In: *Journal of Heat Transfer* 143 (2021) Nr. 6. ISSN: 0022-1481. DOI: 10.1115/1.4050542.

[Car19a] Carleo, G. ; Cirac, I. ; Cranmer, K. ; Daudet, L. ; Schuld, M. ; Tishby, N. ; Vogt-Maranto, L. ; Zdeborová, L.: Machine learning and the physical sciences. In: *Reviews of Modern Physics* 91 (2019) Nr. 4. ISSN: 0034-6861. DOI: 10.1103/RevModPhys.91.045002.

[Car19b] Carvalho, T. P. ; Soares, F. A. A. M. N. ; Vita, R. ; Da Francisco, R. P. ; Basto, J. P. ; Alcalá, S. G. S.: A systematic literature review of machine learning methods applied to predictive maintenance. In: *Computers & Industrial Engineering* 137 (2019), p. 106024. ISSN: 0360-8352. DOI: 10.1016/j.cie.2019.106024. URL: https://www.sciencedirect.com/science/article/pii/s0360835219304838.

[Cha95] Chauvin, Y. ; Rumelhart, D. E.: Backpropagation: Theory, architectures, and applications. Developments in connectionist theory Back propagation. Hillsdale, N.J.: Lawrence Erlbaum Associates, Inc, (1995). ISBN: 9781134775811.

[Che20] Chen, J. ; Imanian, A. ; Wei, H. ; Iyyer, N. ; Liu, Y.: Piecewise stochastic rainflow counting for probabilistic linear and nonlinear damage accumulation considering loading and material uncertainties. In: *International Journal of Fatigue* 140 (2020), p. 105842. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2020.105842. URL: https://www.sciencedirect.com/science/article/pii/s014211232030373x.

[Cla15] Claesen, M. ; Moor, B. de: Hyperparameter Search in Machine Learning. (2015). URL: https://arxiv.org/pdf/1502.02127.

[Cuo22] Cuomo, S. ; Di Cola, V. S. ; Giampaolo, F. ; Rozza, G. ; Raissi, M. ; Piccialli, F.: Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. In: *Journal of Scientific Computing* 92 (2022) Nr. 3, pp. 1–62. ISSN: 1573-7691. DOI: 10.1007/s10915-022-01939-z. URL: https://link.springer.com/article/10.1007/s10915-022-01939-z.

[Das22] Das, D. ; Cen, Y. ; Wang, J. ; Fong, X.: Design of Spintronics-based Neuronal and Synaptic Devices for Spiking Neural Network Circuits. In: (2022). URL: https://arxiv.org/pdf/2211.06630.

[Dei22] Deiana, A. M. ; Tran, N. ; Agar, J. ; Blott, M. ; Di Guglielmo, G. ; Duarte, J. ; Harris, P. ; Hauck, S. ; Liu, M. ; Neubauer, M. S. ; Ngadiuba, J. ; Ogrenci-Memik, S. ; Pierini, M. ; Aarrestad, T. ; Bähr, S. ; Becker, J. ; Berthold, A.-S. ; Bonventre, R. J. ; Müller Bravo, T. E. ; Diefenthaler, M. ; Dong, Z. ; Fritzsche, N. ; Gholami, A. ; Govorkova, E. ; Guo, D. ; Hazelwood, K. J. ; Herwig, C. ; Khan, B. ; Kim, S. ; Klijnsma, T. ; Liu, Y. ; Lo, K. H. ; Nguyen, T. ; Pezzullo, G. ; Rasoulinezhad, S. ; Rivera, R. A. ; Scholberg, K. ; Selig, J. ; Sen, S. ; Strukov, D. ; Tang, W. ; Thais, S. ; Unger, K. L. ; Vilalta, R. ; Krosigk, B. von ; Wang, S. ; Warburton, T. K.: Applications and Techniques for Fast Machine Learning in Science. In: *Frontiers in Big Data* 5 (2022), p. 787421. ISSN: 2624-909X. DOI: 10.3389/fdata.2022.787421. URL: https://www.frontiersin.org/articles/10.3389/fdata.2022.787421/full.

[Deu18] Deutsch, J. ; He, D.: Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48 (2018) Nr. 1, pp. 11–20. ISSN: 2168-2216. DOI: 10.1109/tsmc.2017.2697842.

[Din18] Ding, B. ; Qian, H. ; Zhou, J.: Activation functions and their characteristics in deep neural networks. In: *2018 Chinese Control And Decision Conference (CCDC)*. IEEE, (2018). DOI: 10.1109/ccdc.2018.8407425.

[Div23] Divya, D. ; Marath, B. ; Santosh Kumar, M. B.: Review of fault detection techniques for predictive maintenance. In: *Journal of Quality in Maintenance Engineering* 29 (2023) Nr. 2, pp. 420–441. ISSN: 1355-2511. DOI: 10.1108/JQME-10-2020-0107. URL: https://www.emerald.com/insight/content/doi/10.1108/jqme-10-2020-0107/full/pdf.

[Don08] Dong, P. ; Hong, J. K.: The Master S-N Curve Approach to Fatigue Evaluation of Offshore and Marine Structures. In: *ASME 2004 23rd International Conference on Offshore Mechanics and Arctic Engineering* (2008), pp. 847–855. DOI: 10.1115/OMAE2004-51324.

[Fac18] Facchinetti, M. L.: Fatigue damage of materials and structures assessed by Wöhler and Gassner frameworks: recent insights about load spectra for the automotive. In: *Procedia Engineering* 213 (2018), pp. 117–125. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2018.02.013. URL: https://www.sciencedirect.com/science/article/pii/s1877705818302352.

[Fu17] Fu, W. ; Menzies, T.: Easy over hard: a case study on deep learning. In: Bodden, E. (ed.): *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM Digital Library. New York, NY: ACM, (2017), pp. 49–60. ISBN: 9781450351058. DOI: 10.1145/3106237.3106256.

[Gao14] Gao, H. ; Huang, H.-Z. ; Zhu, S.-P. ; Li, Y.-F. ; Yuan, R.: A modified nonlinear damage accumulation model for fatigue life prediction considering load interaction effects. In: *TheScientificWorldJournal* 2014 (2014), p. 164378. DOI: 10.1155/2014/164378. URL: https://www.hindawi.com/journals/tswj/2014/164378/.

[Goo16] Goodfellow, I. ; Bengio, Y. ; Courville, A.: Deep learning. Adaptive computation and machine learning. Cambridge, Massachusetts and London, England: The MIT Press, (2016). ISBN: 9780262035613.

[Goy20] Goyal, M. ; Goyal, R. ; Venkatappa Reddy, P. ; Lall, B.: Activation Functions. In: (2020), pp. 1–30. DOI: 10.1007/978-3-030-31760-7{\textunderscore}1. URL: https://link.springer.com/chapter/10.1007/978-3-030-31760-7_1.

[Gro13] Grossberg, S.: Recurrent neural networks. In: *Scholarpedia* 8 (2013) Nr. 2, p. 1888. ISSN: 1941-6016. DOI: 10.4249/scholarpedia.1888. URL: http://scholarpedia.org/article/recurrent_neural_network.

[Guo20] Guo, Y. ; Cao, X. ; Liu, B. ; Gao, M.: Solving Partial Differential Equations Using Deep Learning and Physical Constraints. In: *Applied Sciences* 10 (2020) Nr. 17, p. 5917. DOI: 10.3390/app10175917. URL: https://www.mdpi.com/808808.

[Hai06] Haibach, E.: Betriebsfestigkeit: Verfahren und Daten zur Bauteilberechnung. 3. Auflage. VDI-Buch. Berlin, Heidelberg: Springer Berlin Heidelberg, (2006). ISBN: 9783540293644. DOI: Erwin.

[Ham20] Hamilton, J. D.: Time Series Analysis. Princeton, NJ: Princeton University Press, (2020). ISBN: 9780691218632. DOI: 10.1515/9780691218632.

[Han01] Hanumanna, D. ; Narayanan, S. ; Krishnamurthy, S.: Bending fatigue testing of gear teeth under random loading. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 215 (2001) Nr. 7, pp. 773–784. ISSN: 0954-4062. DOI: 10.1243/0954406011524135.

[He21] He, B. ; Liu, L. ; Zhang, D.: Digital Twin-Driven Remaining Useful Life Prediction for Gear Performance Degradation: A Review. In: *Journal of Computing and Information Science in Engineering* 21 (2021) Nr. 3. ISSN: 1530-9827. DOI: 10.1115/1.4049537.

[Hel20] Helm, J. M. ; Swiergosz, A. M. ; Haeberle, H. S. ; Karnuta, J. M. ; Schaffer, J. L. ; Krebs, V. E. ; Spitzer, A. I. ; Ramkumar, P. N.: Machine Learning and Artificial Intelligence: Definitions, Applications, and Future Directions. In: *Current reviews in musculoskeletal medicine* 13 (2020) Nr. 1, pp. 69–76. ISSN: 1935-973X. DOI: 10.1007/s12178-020-09600-8.

[HEU05] HEULER, P. ; KLATSCHKE, H.: Generation and use of standardised load spectra and load–time histories. In: *International Journal of Fatigue* 27 (2005) Nr. 8, pp. 974–990. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2004.09.012. URL: https://www.sciencedirect.com/science/article/pii/s014211230500071x.

[Hua20] Huang, Z.: Classification of Large-Scale High-Resolution SAR Images with Deep Transfer Learning. Institute of Electrical and Electronics Engineers (IEEE), (2020). DOI: 10.36227/techrxiv.11474400.

[Iwa21] Iwana, B. K. ; Uchida, S.: An empirical survey of data augmentation for time series classification with neural networks. In: *PLOS ONE* 16 (2021) Nr. 7, e0254841. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0254841. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0254841.

[Jab14] Jabbar, H. K. ; Khan, R. Z.: Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study). In: *Computer Science, Communication and Instrumentation Devices*. Singapore: Research Publishing Services, (2014). DOI: 10.3850/978-981-09-5247-1{\textunderscore}017.

[Jai99] Jain, L. C. (ed.): Recurrent neural networks: Design and applications. 0th ed. The CRC Press international series on computational intelligence Recurrent neural networks. CRC Press, (1999). ISBN: 9781003040620. DOI: 10.1201/9781420049176.

[Jan21] Janiesch, C. ; Zschech, P. ; Heinrich, K.: Machine learning and deep learning. In: *Electronic Markets* 31 (2021) Nr. 3, pp. 685–695. ISSN: 1019-6781. DOI: 10.1007/s12525-021-00475-2.

[Jan82] Jan Ove Holmen: Fatigue of Concrete by Constant and Variable Amplitude loading. In: *Special Publication* 75 (1982), pp. 71–110. DOI: 10.14359/6402.

[Jay11] Jayalakshmi, T. ; Santhakumaran, A.: Statistical Normalization and Back Propagationfor Classification. In: *International Journal of Computer Theory and Engineering* 3 (2011) Nr. 1, pp. 89–93. ISSN: 1793-8201. DOI: 10.7763/IJCTE.2011.V3.288. URL: https://www.researchgate.net/profile/santhakumaran-a/publication/260024206_statistical_normalization_and_back_propagation_for_clasification.

[Kan18] Kannojia, S. P. ; Jaiswal, G.: Effects of Varying Resolution on Performance of CNN based Image Classification An Experimental Study. In: *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING* 6 (2018) Nr. 9, pp. 451–456. ISSN: 2347-2693. DOI: 10.26438/ijcse/v6i9.451456. URL: https://www.researchgate.net/profile/gaurav-jaiswal-5/publication/328960034_effects_of_varying_resolution_on_performance_of_cnn_based_image_classification_an_experimental_study.

[Kar20] Karabacak, Y. ; Gürsel Özmen, N. ; Gümüşel, L.: Worm gear condition monitoring and fault detection from thermal images via deep learning method. In: *Eksploatacja i Niezawodność – Maintenance and Reliability* 22 (2020) Nr. 3, pp. 544–556. ISSN: 1507-2711. DOI: 10.17531/ein.2020.3.18.

[Kar22] Karabacak, Y. E. ; Gürsel Özmen, N.: Common spatial pattern-based feature extraction and worm gear fault detection through vibration and acoustic measurements. In: *Measurement* 187 (2022), p. 110366. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2021.110366. URL: https://www.sciencedirect.com/science/article/pii/s0263224121012604.

[Kle19] Klein, B. ; Gänsicke, T.: Schwingbeanspruchte Strukturen. In: *Leichtbau-Konstruktion*. Springer Vieweg, Wiesbaden, (2019), pp. 387–424. DOI: 10.1007/978-3-658-26846-6{\textunderscore}24. URL: https://link.springer.com/chapter/10.1007/978-3-658-26846-6_24.

[LeC15] LeCun, Y. ; Bengio, Y. ; Hinton, G.: Deep learning. In: *Nature* 521 (2015) Nr. 7553, pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://www.nature.com/articles/nature14539.

[Lee11] Lee, Y.-L.: Fatigue Testing and Analysis: Theory and Practice. Burlington: Elsevier Science, (2011). ISBN: 9780750677196.

[Li15] Li, W. ; Mo, W. ; Zhang, X. ; Squiers, J. J. ; Lu, Y. ; Sellke, E. W. ; Fan, W. ; DiMaio, J. M. ; Thatcher, J. E.: Outlier detection and removal improves accuracy of machine learning approach to multispectral burn diagnostic imaging. In: *Journal of Biomedical Optics* 20 (2015) Nr. 12, p. 121305. ISSN: 1560-2281. DOI: 10.1117/1.JBO.20.12.121305. URL: https://www.spiedigitallibrary.org/journals/journal-of-biomedical-optics/volume-20/issue-12/121305/outlier-detection-and-removal-improves-accuracy-of-machine-learning-approach/10.1117/1.jbo.20.12.121305.short.

[Lit81] Little, R. E. ; Ekvall, J. C.: Statistical Analysis of Fatigue Data: A Symposium. ASTM International, (1981).

[Lu23] Lu, S. ; Lu, J. ; An, K. ; Wang, X. ; He, Q.: Edge Computing on IoT for Machine Signal Processing and Fault Diagnosis: A Review. In: *IEEE Internet of Things Journal* (2023), p. 1. ISSN: 2327-4662. DOI: 10.1109/jiot.2023.3239944.

[Luo16] Luo, G.: A review of automatic selection methods for machine learning algorithms and hyper-parameter values. In: *Network Modeling Analysis in Health Informatics and Bioinformatics* 5 (2016) Nr. 1, pp. 1–16. ISSN: 2192-6670. DOI: 10.1007/s13721-016-0125-6. URL: https://link.springer.com/article/10.1007/s13721-016-0125-6.

[Lv15] Lv, Z. ; Huang, H.-Z. ; Zhu, S.-P. ; Gao, H. ; Zuo, F.: A modified nonlinear fatigue damage accumulation model. In: *International Journal of Damage Mechanics* 24 (2015) Nr. 2, pp. 168–181. ISSN: 1056-7895. DOI: 10.1177/1056789514524075.

[Lyu19] Lyu, Y. ; Bai, L. ; Huang, X.: ChipNet: Real-Time LiDAR Processing for Drivable Region Segmentation on an FPGA. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66 (2019) Nr. 5, pp. 1769–1779. ISSN: 1549-8328. DOI: 10.1109/TCSI.2018.2881162.

[Med19] Medina, R. ; Cerrada, M. ; Cabrera, D. ; Sanchez, R.-V. ; Li, C. ; Oliveira, J. V. de: Deep Learning-Based Gear Pitting Severity Assessment Using Acoustic Emission, Vibration and Currents Signals. In: *2019 Prognostics and System Health Management Conference (PHM-Paris)*. IEEE, (2019). DOI: 10.1109/phm-paris.2019.00042.

[Mik18] Mikolajczyk, A. ; Grochowski, M.: Data augmentation for improving deep learning in image classification problem. In: *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. IEEE, (2018). DOI: 10.1109/iiphdw.2018.8388338.

[Mil77] Miller, K. J. ; Zachariah, K. P.: Cumulative damage laws for fatigue crack initiation and stage i propagation. In: *The Journal of Strain Analysis for Engineering Design* 12 (1977) Nr. 4, pp. 262–270. ISSN: 0309-3247. DOI: 10.1243/03093247v124262.

[Min45] Miner, M. A.: Cumulative Damage in Fatigue. In: *Journal of Applied Mechanics* 12 (1945) Nr. 3, A159–A164. ISSN: 0021-8936. DOI: 10.1115/1.4009458.

[Min61] Minsky, M.: Steps toward artificial intelligence. In: *Proceedings of the IRE* (1961) Nr. 1, pp. 8–30. URL: https://courses.csail.mit.edu/6.803/pdf/steps.pdf.

[Mis20] Misyris, G. S. ; Venzke, A. ; Chatzivasileiadis, S.: Physics-Informed Neural Networks for Power Systems. In: *2020 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, (2020). DOI: 10.1109/pesgm41954.2020.9282004.

[Mob02] Mobley, R. K.: Introduction to Predictive Maintenance. 2nd ed. Plant Engineering Ser. Oxford: Elsevier Science & Technology, (2002). ISBN: 9780080478692.

[Mur12] Murphy, K. P.: Machine learning: A probabilistic perspective. Adaptive computation and machine learning series. Cambridge, Mass.: MIT Press, (2012). ISBN: 978-0262018029.

[Muz19] Muzaffar, S. ; Afshari, A.: Short-Term Load Forecasts Using LSTM Networks. In: *Energy Procedia* 158 (2019), pp. 2922–2927. ISSN: 1876-6102. DOI: 10.1016/j.egypro.2019.01.952. URL: https://www.sciencedirect.com/science/article/pii/s1876610219310008.

[Nic13] Nicholas Bugliarello, Biji George, Don Giessel, Dan McCurdy, Ron Perkins, Steve Richardson, and Craig Zimmerman: Heat Treat Process for Gears. (2013). URL: http://thermalprocessing.com/media/uploads/assets/pdf/articles/2013_spring/2013_bodycote.pdf.

[Par20] Park, J.-Y. ; Hwang, Y. ; Lee, D. ; Kim, J.-H.: MarsNet: Multi-Label Classification Network for Images of Various Sizes. In: *IEEE Access* 8 (2020), pp. 21832–21846. ISSN: 2169-3536. DOI: 10.1109/access.2020.2969217.

[Pav17] Pavlo M. Radiuk: Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. In: *Information Technology and Management Science* 20 (2017) Nr. 1, pp. 20–24. ISSN: 2255-9094. URL: https://itms-journals.rtu.lv/article/view/itms-2017-0003.

[Pav18] Pavlou, D. G.: The theory of the S-N fatigue damage envelope: Generalization of linear, double-linear, and non-linear fatigue damage models. In: *International Journal of Fatigue* 110 (2018), pp. 204–214. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2018.01.023. URL: https://www.sciencedirect.com/science/article/pii/s014211231830029x.

[Pes14] Peshawa J Muhammad Ali ; Rezhna Hassan Faraj: Data Normalization and Standardization: A Technical Report. Unpublished, (2014). DOI: 10.13140/RG.2.2.28948.04489. URL: https://www.researchgate.net/profile/peshawa-muhammad-ali/publication/340579135_data_normalization_and_standardization_a_technical_report.

[Phi19] Philipp Probst, Anne-Laure Boulesteix, Bernd Bischl: Tunability: Importance of hyperparameters of machine learning algorithms. (2019). URL: https://www.jmlr.org/papers/volume20/18-444/18-444.pdf.

[Pra14] Praveenkumar, T. ; Saimurugan, M. ; Krishnakumar, P. ; Ramachandran, K. I.: Fault Diagnosis of Automobile Gearbox Based on Machine Learning Techniques. In: *Procedia Engineering* 97 (2014), pp. 2092–2098. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2014.12.452. URL: https://www.sciencedirect.com/science/article/pii/s1877705814035022x.

[Pun06] Pungo, N. ; CIAVARELLA, M. ; CORNETTI, P. ; CARPINTERI, A.: A generalized Paris' law for fatigue crack growth. In: *Journal of the Mechanics and Physics of Solids* 54 (2006) Nr. 7, pp. 1333–1349. ISSN: 0022-5096. DOI: 10.1016/j.jmps.2006.01.007. URL: https://www.sciencedirect.com/science/article/pii/s0022509606000196.

[Raj20] Raju, V. N. G. ; Lakshmi, K. P. ; Jain, V. M. ; Kalidindi, A. ; Padma, V.: Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification. In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, (2020). DOI: 10.1109/icssit48917.2020.9214160.

[Rao96] Rao, B. K. N.: Handbook of condition monitoring. 1. ed. Oxford: Elsevier Advanced Technology, (1996). ISBN: 9781856172349.

[Reg17] Rege, K. ; Pavlou, D. G.: A one-parameter nonlinear fatigue damage accumulation model. In: *International Journal of Fatigue* 98 (2017), pp. 234–246. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2017.01.039. URL: https://www.sciencedirect.com/science/article/pii/s014211231730049x.

[Sai15] Sainath, T. N. ; Vinyals, O. ; Senior, A. ; Sak, H.: Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. (2015). DOI: 10.1109/icassp.2015.7178838.

[Sal17] Salehinejad, Hojjat and Sankar, Sharan and Barfett, Joseph and Colak, Errol and Valaee, Shahrokh: Recent advances in recurrent neural networks. In: (2017). URL: https://arxiv.org/pdf/1801.01078.pdf.

[San18] Sander, M.: Sicherheit und Betriebsfestigkeit von Maschinen und Anlagen: Konzepte und Methoden zur Lebensdauervorhersage. 2., aktualisierte und ergänzte Auflage. Berlin, Germany and Heidelberg: Springer Vieweg, (2018). ISBN: 9783662544426. DOI: 10.1007/978-3-662-54443-3.

[Sho19] Shorten, C. ; Khoshgoftaar, T. M.: A survey on Image Data Augmentation for Deep Learning. In: *Journal of big data* 6 (2019) Nr. 1. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0.

[Sin20] Singh, D. ; Singh, B.: Investigating the impact of data normalization on classification performance. In: *Applied Soft Computing* 97 (2020), p. 105524. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2019.105524. URL: https://www.sciencedirect.com/science/article/pii/s1568494619302947.

[Sko99] Skorupa, M.: Load interaction effects during fatigue crack growth under variable amplitude loading-a literature review. Part II: qualitative interpretation. In: *Fatigue & Fracture of Engineering Materials & Structures* 22 (1999) Nr. 10, pp. 905–926. ISSN: 1460-2695. DOI: 10.1046/j.1460-2695.1999.00158.x.

[Sta19] Staudemeyer, R. C. ; Morris, E. R.: Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. (2019).

[Stu10] Stuart J. Russell, Peter Norvig ; Russell, S. J. ; Norvig, P.: Artificial Intelligence - A Modern Approach // Artificial intelligence: A modern approach: Third Edition. 3. ed. Prentice-Hall series in artificial intelligence. Upper Saddle River, NJ: Prentice-Hall, (2010). ISBN: 978-0-13-604259-4. DOI: Stuart.

[Sub76] Subramanyan, S.: A Cumulative Damage Rule Based on the Knee Point of the S-N Curve. In: *Journal of Engineering Materials and Technology* 98 (1976) Nr. 4, pp. 316–321. ISSN: 0094-4289. DOI: 10.1115/1.3443383.

[Sun14] Sun, Q. ; Dui, H.-N. ; Fan, X.-L.: A statistically consistent fatigue damage model based on Miner's rule. In: *International Journal of Fatigue* 69 (2014), pp. 16–21. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2013.04.006. URL: https://www.sciencedirect.com/science/article/pii/s0142112313001047.

[Sut20] Sutton, R. S. ; Barto, A.: Reinforcement learning: An introduction. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts and London, England: The MIT Press, (2020). ISBN: 9780262039246.

[Tak22] Takuma Seno, M. I.: d3rlpy: An Offline Deep Reinforcement Learning Library. Journal of Machine Learning Research 23, (2022). URL: https://arxiv.org/pdf/2111.03788.pdf.

[Tay18] Taylor, L. ; Nitschke, G.: Improving Deep Learning with Generic Data Augmentation. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, (2018). DOI: 10.1109/ssci.2018.8628742.

[The15] Theodoridis, S.: Machine Learning: A Bayesian and Optimization Perspective. NET Developers Series. Burlington: Elsevier Science, (2015). ISBN: 9780128015223. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=2007481.

[Tho20] Thompson, N. C. ; Greenewald, K. ; Lee, K. ; Manso, G. F.: The Computational Limits of Deep Learning. In: (2020). URL: https://arxiv.org/pdf/2007.05558.

[Tho22] Thommandru, A. ; Mutkule, P. ; Bandi, A. ; Tongkachok, K.: Towards Applicability of Artificial Intelligence in Healthcare, Banking and Education Sector. In: *ECS Transactions* 107 (2022) Nr. 1, pp. 16665–16671. ISSN: 1938-5862. DOI: 10.1149/10701.16665ecst.

[Tiw09] Tiwari ; N Tanwar ; S Mishra: APPLICATION OF SUPPORT VECTOR MACHINE TECHNIQUES FOR HEALTH MONITORING OF GEARS. Condition Monitoring Society of India, (2009). URL: https://www.researchgate.net/profile/rajiv-tiwari/publication/357418196_application_of_support_vector_machine_techniques_for_health_monitoring_of_gears.

[Ver21] Verbraeken, J. ; Wolting, M. ; Katzy, J. ; Kloppenburg, J. ; Verbelen, T. ; Rellermeyer, J. S.: A Survey on Distributed Machine Learning. In: *ACM Computing Surveys* 53 (2021) Nr. 2, pp. 1–33. ISSN: 0360-0300. DOI: 10.1145/3377454.

[Vie20] Vietze, D. ; Hein, M. ; Stahl, K.: Method for a cloud based remaining-service-life-prediction for vehicle-gearboxes based on big-data-analysis and machine learning. In: *Forschung im Ingenieurwesen* 84 (2020) Nr. 4, pp. 305–314. ISSN: 1434-0860. DOI: 10.1007/s10010-020-00415-0. URL: https://link.springer.com/article/10.1007/s10010-020-00415-0.

[Vog16] Vogel-Heuser, B.: Handbuch Industrie 4.0 Bd.4: Allgemeine Grundlagen. 2nd ed. VDI Springer Reference. Berlin, Heidelberg: Springer Berlin Heidelberg, (2016). ISBN: 9783662532546. URL: http://gbv.eblib.com/patron/FullRecord.aspx?p=4748961.

[Vul20] Vullo, V.: Gears: Volume 1: Geometric and Kinematic Design. 1st ed. 2020. Vol. 10. Springer Series in Solid and Structural Mechanics. Cham: Springer International Publishing and Imprint Springer, (2020). ISBN: 9783030365028. DOI: 10.1007/978-3-030-36502-8.

[Wan18] Wang, D. ; Tsui, K.-L. ; Miao, Q.: Prognostics and Health Management: A Review of Vibration Based Bearing and Gear Health Indicators. In: *IEEE Access* 6 (2018), pp. 665–676. ISSN: 2169-3536. DOI: 10.1109/access.2017.2774261.

[Wen22] Wen, Q. ; Sun, L. ; Yang, F. ; Song, X. ; Gao, J. ; Wang, X. ; Xu, H.: Time Series Data Augmentation for Deep Learning: A Survey. (2022). DOI: 10.24963/ijcai.2021/631.

[Wer00] Werner, S. ; Bacher-Höchst, M. ; Sonsino, C. M.: Stabilität der tatsächlichen Schadenssumme gegen eine Variation des Kollektivumfangs, der Kerbschärfe und des Kollektivhöchstwertes am Beispiel der warmausgehärteten Knetlegierung AlMgSi1. In: *Materialwissenschaft und Werkstofftechnik* 31 (2000) Nr. 7, pp. 589–597. ISSN: 0933-5137. DOI: 10.1002/1521-4052(200007)31:7{\textless}589::AID-MAWE589{\textgreater}3.0.CO;2-L.

[Wil18] Will Koehrsen: Overfitting vs. underfitting: A complete example. (2018). URL: http://www.pstu.ac.bd/files/materials/1566949131.pdf.

[Wit17] Wittel, H. ; Jannasch, D. ; Voßiek, J. ; Spura, C.: Maschinenelemente. 23., überarbeitete und erweiterte Auflage. Wiesbaden and Heidelberg: Springer Vieweg, (2017). ISBN: 9783658178956. DOI: 10.1007/978-3-658-17896-3.

[Wue16] Wuest, T. ; Weimer, D. ; Irgens, C. ; Thoben, K.-D.: Machine learning in manufacturing: advantages, challenges, and applications. In: *Production & Manufacturing Research* 4 (2016) Nr. 1, pp. 23–45. DOI: 10.1080/21693277.2016.1192517.

[Xia16] Xiaogang Wang, Z Lei, X Zhang, B Zhou, J Peng: Machine learning basics: Deep learning. (2016). URL: http://whdeng.cn/teaching/ppt_01_machine%20learning%20basics.pdf.

[Yan19] Yang, J. ; Rahardja, S. ; Fränti, P.: Outlier detection. In: *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*. New York, NY, USA: ACM, (2019). DOI: 10.1145/3371425.3371427.

[Yan20a] Yan, H. ; Qin, Y. ; Xiang, S. ; Wang, Y. ; Chen, H.: Long-term gear life prediction based on ordered neurons LSTM neural networks. In: *Measurement* 165 (2020), p. 108205. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2020.108205. URL: https://www.sciencedirect.com/science/article/pii/s0263224120307430.

[Yan20b] Yang, L. ; Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. In: *Neurocomputing* 415 (2020), pp. 295–316. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.07.061. URL: https://www.sciencedirect.com/science/article/pii/s0925231220311693.

[Yos03] Yoshua Bengio , Yves Grandvalet: No unbiased estimator of the variance of k-fold cross-validation. MIT Press, (2003). URL: https://proceedings.neurips.cc/paper_files/paper/2003/file/e82c4b19b8151ddc25d4d93baf7b908f-Paper.pdf.

[Yuk04] Yuksel, C. ; Kahraman, A.: Dynamic tooth loads of planetary gear sets having tooth profile wear. In: *Mechanism and Machine Theory* 39 (2004) Nr. 7, pp. 695–715. ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2004.03.001. URL: https://www.sciencedirect.com/science/article/pii/s0094114x04000448.

[Zha16] Zhao, R. ; Wang, J. ; Yan, R. ; Mao, K.: Machine health monitoring with LSTM networks. (2016). DOI: 10.1109/icsenst.2016.7796266.

[Zha17] Zhang, C. ; Bengio, S. ; Hardt, M. ; Recht, B. ; Vinyals, O.: Understanding deep learning requires rethinking generalization. (2017). DOI: ICLR. URL: https://arxiv.org/pdf/1611.03530.

[Zha23] Zhang, X. ; Liu, C.-A.: Model averaging prediction by K -fold cross-validation. In: *Journal of Econometrics* 235 (2023) Nr. 1, pp. 280–301. ISSN: 0304-4076. DOI: 10.1016/j.jeconom.2022.04.007. URL: https://www.sciencedirect.com/science/article/pii/s0304407622000975.

[Zhu05] Zhu, X.: Semi-Supervised Learning Literature Survey. In: (2005). URL: https://minds.wisconsin.edu/handle/1793/60444.

[Zhu19] Zhu, S.-P. ; Liao, D. ; Liu, Q. ; Correia, J. A. ; Jesus, A. M. de: Nonlinear fatigue damage accumulation: Isodamage curve-based model and life prediction aspects. In: *International Journal of Fatigue* 128 (2019), p. 105185. ISSN: 0142-1123. DOI: 10.1016/j.ijfatigue.2019.105185. URL: https://www.sciencedirect.com/science/article/pii/s0142112319302750.

[Zuo15] Zuo, F.-J. ; Huang, H.-Z. ; Zhu, S.-P. ; Lv, Z. ; Gao, H.: Fatigue life prediction under variable amplitude loading using a non-linear damage accumulation model. In: *International Journal of Damage Mechanics* 24 (2015) Nr. 5, pp. 767–784. ISSN: 1056-7895. DOI: 10.1177/1056789514553042.

Webpages

[Gil23] Gilbert Tanner: Activation Functions: Acces Date: 21.05.2023. (2023). URL: https://ml-explained.com/blog/activation-functions-explained.

[Goo23] Google: Machine Learning Glossary: Acces Date: 20/05/2023. (2023). URL: https://developers.google.com/machine-learning/glossary?hl=de.

[sci23] scikit: scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation, Access date: 25/06/2023. (2023). URL: https://scikit-learn.org/stable/index.html.

[SOL23] SOLIDWORKS: Derivation of Basquin Constants from S-N curve - 2020 - SOLIDWORKS Help (Access Date: 19/06/2023). (2023). URL: https://help.solidworks.com/2020/english/SolidWorks/cworks/r_Calculation_Basquin_Constants_S-N_curve.htm .

[Sym23] Symbrium | An Engineer's Engineering Company: Symbrium Single Tooth Bending Gear Testing Systems: Access date: 17.06.2023. (2023). URL: https://www.symbrium.com/symbrium-single-tooth-bending-gear-testing-systems/.

[tsl23] tslearn: tslearn's documentation — tslearn 0.5.3.2 documentation, Access date: 25/06/2023. (2023). URL: https://tslearn.readthedocs.io/en/stable/#.

[XGB23] XGBoost: XGBoost Documentation — xgboost 1.7.6 documentation (Access Date: 02/07/2023). (2023). URL: https://xgboost.readthedocs.io/en/stable/.

List of Figures

List of Tables