Universidade do Minho
Escola de Engenharia

Renan Belli Berman

Developing of Computing Models for Defect Identification in Wind Turbine Blades

FRP++
Advanced structural analysis and design using composite materials

Funded by the European Union

September 2023

Developing of Computing Models for Defect Identification in Wind Turbine Blades
Renan Belli Berman
FRP++ Advanced structural analysis and design using composite materials
UMinho | 2023

The European Master in Advanced Structural Analysis and Design using Composite Materials is a join initiative of:

Universidade do Minho · Universitat de Girona · Università degli Studi di Napoli Federico II · INSA Institut National des Sciences Appliquées Toulouse · Université Toulouse III Paul Sabatier

Universidade do Minho
Escola de Engenharia

Renan Belli Berman

Developing of Computing Models for Defect Identification in Wind Turbine Blades

FRP++
Advanced structural analysis and design using composite materials

Master Dissertation
European Master Advanced Structural Analysis and Design using Composite Materials

Work developed under the supervision of
José Sena Cruz
Serena Langiano

Funded by the European Union

September 2023

DECLARATION

Name: Renan Belli Berman

Title of the Thesis: Developing of Computing Models for Defect Identification in Wind Turbine Blades

Supervisors:

José Sena Cruz

Serena Langiano

Year of Conclusion: 2023

Master Course in Advanced Structural Analysis and Design using Composite Materials

University of Minho, 04/09/2023

Signature:

ACKNOWLEDGEMENTS

First, I would like to thank the directors of the FRP++ master course for this opportunity. This master course would not be possible without your hard work.

I would also like to thank the Erasmus Mundus project, the scholarship provided was fundamental for my full dedication during the development of this work, and thus indispensable for the knowledge acquired and the lessons learned during the execution of this master.

I would like to give thanks especially to the University of Girona and the University of Minho for the warm welcome, the helpful attitude of everyone and all the knowledge shared.

I would like to thank my supervisor Prof. José Sena Cruz for this opportunity, for all the knowledge he has shared during the writing of this thesis and for his unfaltering good humour.

I would also like to thank my co-supervisor Serena Langiano for her support and for the many helpful suggestions on the data augmentation process, as well as the obtaining of a supplementing dataset.

Thanks to my mother, Andrea Cristina Belli, her unwavering trust in me and the strength of her will are a source of inspiration.

Thanks to my brother, João Vitor Belli, his rationality, intelligence and honesty are a constant source of reassurance.

Thanks to my friend and teacher, Hazim Ali Al-Qureshi, for all the advice and patience when hearing about my complaints and difficulties.

Thanks to all my family for the constant cheer and support, all my accomplishments would be impossible without you.

Thanks to my friends Marcelo Demetrio de Magalhães and Paulo Vitor Carvalho, for the late nights talking about the future, the weekends spent watching horrible movies and the fights over irrelevant details that never failed to lift my mood.

Desenvolvimento de modelos computacionais para a deteção de dano em turbinas eólicas

RESUMO

A energia eólica, com sua rápida expansão global, destaca-se como uma das fontes de energia renovável de crescimento mais rápido. Com os avanços na tecnologia das turbinas eólicas, houve um aumento notável no tamanho das pás, melhorando sua eficácia na captação do vento. O setor da energia eólica está ativamente explorando tecnologias inovadoras de inspeção para mitigar riscos humanos e minimizar custos. Com os avanços em robótica e técnicas de inspeção, algoritmos de aprendizado automática e inteligência artificial estão sendo desenvolvidos para automatizar e processar eficientemente grandes volumes de dados, reduzindo custos e riscos de segurança relacionados ao perigoso trabalho humano necessário para inspecionar as pás das turbinas eólicas. Em linha com esse objetivo, a presente dissertação de mestrado visa desenvolver e/ou melhorar uma ferramenta de inteligência artificial capaz de processar e detetar defeitos em imagens RGB das pás de turbinas eólicas. Conclui-se que as redes neurais convolucionais se destacam no processo de classificação de imagens em grande escala. Consequentemente, um modelo de rede neural convolucional é proposto e posteriormente treinado. Três diferentes técnicas de treino são exploradas e comparadas na capacidade de previsão geral do modelo final. Apesar do desempenho promissor para bancos de dados mais volumosos, o modelo não conseguiu treinar efetivamente no conjunto de dados de danos das pás de turbina disponível, principalmente devido ao tamanho limitado do banco de dados. Uma tentativa subsequente de desenvolver outra arquitetura CNN adaptada para bancos de dados menores também produziu resultados insatisfatórios.

PALAVRAS-CHAVE: Energia Eólica; Pás de turbina eólica; Compósitos reforçados por fibras; Aprendizado automatizado; Redes neurais convolucionais.

Developing of Computing Models for Defect Identification in Wind Turbine Blades

ABSTRACT

Wind energy, with its rapid global expansion, stands as one of the fastest-growing renewable energy sources. As advancements in wind turbine technology unfold, there has been a notable increase in blade sizes, enhancing their wind-capturing efficacy. The wind energy sector is actively exploring innovative inspection technologies to mitigate human risk and minimize costs. With advancements in robotics and inspection techniques, machine learning algorithms and artificial intelligence are being developed to automate and efficiently process significant volumes of data while reducing costs and safety risk related to the hazardous human labor required to inspect the wind turbine blades. In line with this objective, this master's dissertation aims at developing and/or improving an artificial intelligence tool capable of accurately processing and detecting surface in RGB images of wind turbine blades (WTB). An initial exploration of machine learning algorithms was conducted to identify the most suitable techniques for the given task. It was determined that Convolutional Neural Networks (CNN), a branch of machine learning algorithms, are particularly adept at handling large-scale image classification challenges. Consequently, a CNN model or architecture, is proposed and subsequently trained. Three different training techniques are explored and compared on their result in overall prediction capability of the final CNN model upon acquiring the image data, the CNN model underwent training, followed by an evaluation of its predictive accuracy. Despite promising performance, the model was unable to train effectively on the WTB damage dataset, primarily due to the limited size of database. A subsequent attempt to devise another CNN architecture tailored for smaller databases also yielded unsatisfactory results.

KEYWORDS: Wind Energy; Wind Turbine Blades; Fiber Reinforced Composites; Machine Learning; Convoluted Neural Networks.

TABLE OF CONTENTS

Acknowledgements III
Resumo IV
Abstract V
Table of Contents VI
List of Figures VIII
List of Tables X
List of Abbreviations and Symbols XI
1. Introduction 1
1.1. Motivation 1
1.2. Objectives 2
1.3. Structure of the Dissertation 2
2. State of the art 3
2.1. Wind Energy 3
2.1.1. Wind Turbine Towers 5
2.2. Wind Turbine Blades 8
2.3. Inspection Methods for WTBs 12
2.4. Machine Learning and Neural Networks 21
2.5. Convoluted Neural Networks 28
2.6. Summary 42
3. Implementation of the blade damage detection model 45
3.1. Data 47
3.2. Convoluted neural network architecture 50
4. Results and analysis 59
4.1. CNN approach and hyperparameter selection 59
4.2. Results 62
5. Conclusions and further developments 65
5.1. Main Conclusions 65
5.2. Further Development 66

References 67

LIST OF FIGURES

Figure 2.1: Renewable energy installed capacity by year in the world. Source: [7] 4
Figure 2.2: Global wind flows. Source: [8] 5
Figure 2.3: Main wind turbine components. Source: [13] 6
Figure 2.4: Wind blade elements. Source: [14] 7
Figure 2.5: Types of composites based on the form of reinforcement. Source: [21] 10
Figure 2.6: Resin transfer moulding process. Source: [27] 11
Figure 2.7: Vacuum infusion process source: [27] 12
Figure 2.8: Defects of in service use detectable via visual inspection. Source: [16] 14
Figure 2.9: Blistering on the surface of a composite material. Source [28] 15
Figure 2.10: Leading edge erosion of a WTB detectable via visual inspection. Source [29] 16
Figure 2.11: Crack in a WTB detectable via visual inspection. Source: [29] 17
Figure 2.12: Dent in a WTB detectable via visual inspection. Source: [29] 18
Figure 2.13: Shearography of a non-damaged material. Source: [3] 19
Figure 2.14: Traditional pulse thermography setup. Source: [31] 19
Figure 2.15: Thermography of a WTB where the crack growth tip is seen as a bright spot. Source: [32] 20
Figure 2.16: Regression problem where the output is continuous. Source: [36] 22
Figure 2.17: Diagram of a single neuron. Source: Author 23
Figure 2.18: Fully connected layer. Source: Author 24
Figure 2.19: Fully connected multilayer feed-forward neural network. Source: [36] 25
Figure 2.20: Rectified linear activation function. Source: [39] 25
Figure 2.21: Sigmoid activation function. Source: [39] 26
Figure 2.22: Hyperbolic tangent activation function. Source: [39] 27
Figure 2.23: Convolution operation using a 3 × 3 filter and step of one and no padding. Source: [36] 29
Figure 2.24: Example of a convolution operation using a 3 × 3 kernel with padding and a stride of 1. Source: [40] 30
Figure 2.25: Max pooling layer with a 2x2 kernel and stride 2 [36] 30
Figure 2.26: CNN architecture using 2 convoluted layers, 2 pooling layers and two fully connected layers. Source: [38] 31

Figure 2.27: Comparison of prediction error on training and test data with increasing training, source [45] 36
Figure 2.28: Generating new data using data augmentation for rotation, scaling and brightness changes. Source [36] 39
Figure 2.29: Architecture of the ResNet model source [38] 42
Figure 3.1: Training of the example model using the CPU. Source: Author 46
Figure 3.2: Random images from the TensorFlow flowers dataset. Source [44] 48
Figure 3.3: Example of images on the dataset along with the new classification. Source: [28] 49
Figure 3.4: Data augmentation on one of the images of the dataset. Source: [28] 50
Figure 3.5: Test and Validation Accuracy per epoch using the fine-tuning technique. Source: Author 54
Figure 3.6: Test and Validation Accuracy per epoch using the hybrid technique. Source: Author 55
Figure 3.7: Final Architecture of the CNN used for defect detection. Source: Author 56
Figure 3.8: Structure of a single convolutional block in the ResNet50 architecture. Source: Author .. 57

LIST OF TABLES

Table 2.1: Confusion matrix for a binary system 33
Table 2.2: AlexNet architecture. Source [34] 41
Table 3.1: Training of the example model using the GPU 46
Table 3.2: Hyperparameters selected for training. 52
Table 3.3: Results for 15 epochs of training using the transfer learning technique. 53
Table 3.4: Best validation accuracy results for each of the techniques. 55
Table 4.1: Parameters for the grid search executed on the first CNN architecture. 60
Table 4.2: Models trained for the grid search on the first CNN architecture. 60
Table 4.3: Parameters for the grid search executed on the second CNN architecture. 61
Table 4.4: Models trained for the grid search on the second CNN architecture. 61
Table 4.5: Training of Model 1 where omitted epochs were identical. 62
Table 4.6: Training of Model 72 where omitted epochs were identical. 63

List of Abbreviations and Symbols

Abbreviations

WTB	Wind turbine blade
AI	Artificial Intelligence
CNN	Convoluted neural network
RGB	Red Green Blue, used to refer to colour images
RTM	Resin transfer moulding
VIP	Vacuum infusion process
FRP	Fibre Reinforced Polymer
MMC	Metalic matrix composite
CMC	Ceramic matrix composite
PMC	Polymer matrix composite
NDT	Non-destructive testing
ReLU	Rectified linear Unit
Adam	Adaptive moment estimation

Symbols

$\widehat{Y_i}$	Predicted values by the model
Y_i	Actual value of the data point
W_t	Weight parameters of the model at time t
W_{t-1}	Weight parameters of the model at time t - 1
η	Learning rate of the model
$J(W)$	Loss function
$\widehat{m}_t$	Bias corrected moving mean of the gradients
β_1	Momentum hyperparameter
$\hat{v}_t$	Bias corrected moving mean of the squared gradients
β_2	Adaptive learning rate hyperparameter
λ	Weight decay hyperparameter

THIS PAGE WAS INTENTIONALLY LEFT BLANK

1. INTRODUCTION

This chapter introduces the topics involved in this dissertation. The first section elucidates the key themes underpinning this dissertation, grounding its foundational intent and scope. Then the primary aims and objectives that guide the trajectory of this research endeavour are addressed. Concluding this introductory section, to ensure clarity and systematic comprehension of this work, a concise overview of the subsequent chapters and their thematic divisions will be provided.

1.1. Motivation

Globally, wind energy has been experiencing a marked surge, positioning itself as one of the fastest-growing renewable energy sources [1]. Central to this energy conversion process are wind turbine towers, which harness the mechanical energy from the wind and transform it into electricity. The pivotal components in this energy conversion are the wind turbine blades, playing an essential role in capturing the wind's kinetic energy. Over the past two decades, there has been a pronounced shift towards enlarging blade sizes to optimize the kinetic energy extraction. This trend is largely attributed to advancements in composite material fabrication, which allows for the production of more expansive wind turbine blade (WTB) components without compromising their essential mechanical attributes. Nevertheless, the manufacturing of larger blades brings about heightened costs, underscoring the importance of robust inspection and maintenance regimens to ensure their longevity and operational efficiency [2].

The burgeoning growth of the wind energy sector underscores the imperative for innovative inspection methodologies, both to enhance safety by reducing human intervention and to curtail associated costs. Drone imaging has emerged as a promising solution in this landscape [3]. However, one of the principal challenges accompanying drone-based inspections is the volume of data they produce. Evaluating this massive amount of data manually not only intensifies labour demands but also introduces potential inaccuracies and oversights in damage identification. The domain of machine learning, especially the application of convolutional neural networks (CNN), has garnered significant attention owing to its adeptness in processing vast datasets, making it especially apt for automating image detection tasks. In this context, the extensive data generated by drone inspections, which could otherwise be a significant problem, is instead a valuable resource for training and refining these machine learning models [4].

1.2. Objectives

The primary objectives of this master's thesis are to conduct a comprehensive review of existing literature on the state-of-the-art of typical surface and subsurface defects in wind turbine blades (WTB). Additionally, this research aims to analyse the state-of-the-art of currently available, and relevant for this project, image processing models. The research will involve defining and characterizing the parameters for identifying WTB defects based on relevant literature, industry knowledge, and provided samples. Following the literature review and analysis of image processing models, this master's dissertation aims at determining and implementing the optimal image processing algorithm for the available data. Utilizing RGB photographic data provided by EDP NEW, the research will develop a model capable of accurately labelling future images obtained from WTB inspections. This approach will enable the development of an efficient and accurate system for identifying and characterizing WTB defects, contributing to enhanced operational safety and reduced maintenance costs for wind energy systems.

1.3. Structure of the Dissertation

This dissertation is structured into five chapters. Chapter 2 presents a comprehensive review of the current literature, encompassing the domains of wind energy, wind turbines, WTB, prevalent defects observed in WTB, prominent inspection methodologies employed, machine learning, and convolutional neural networks. Chapter 3 delves into the technical aspects, elucidating the hardware deployed for CNN training, predominant programming languages utilized in formulating CNN, foremost libraries available for crafting and training neural networks, the elected architecture for damage detection, intricacies of key training techniques cited in literature, and a detailed overview of the data selected for testing, inclusive of its labelling and augmentation procedures. Chapter 4 articulates the specifics of the designated CNN architecture for training and chronicles the outcomes post-training on the procured WTB inspection data. The final chapter, Chapter 5, encapsulates the findings of the whole work and presents recommendations for prospective research endeavours using this one as a basis.

2. STATE OF THE ART

This chapter introduces the main subjects addressed within this work, supported by the literature review carried out. The chapter starts with an introduction on the general properties of wind energy and its harnessing, followed by wind turbine blades' defects and the usual inspection methods used in the wind energy sector. Next, machine learning and neural networks are covered detailing the main approaches used. Lastly, convoluted neural networks are further detailed along with their main applications.

2.1. Wind Energy

The global demand for electrical energy is increasing with economic and population growth and the normal human daily life is becoming increasingly dependent on energy for multiple aspects such as communication, transportation, health care and leisure. The three major types of fossil fuel, namely coal, crude oil and natural gas, accounted for 86% of the global primary energy production in 2012 [5]. As the awareness of the need for environmental protection has been increasing in the last decades most countries have signed greenhouse reduction agreements such as the Kyoto Protocol or the Paris Agreement. While the first required only a 5 percent reduction in emissions, the latter has the goal of preventing global warming by more than 2 degrees Celsius above preindustrial levels and therefore requires more significant effort from the adhering countries [1], [6]. Therefore, an increasing growth in the demand for renewable energy production is pursued, especially in the solar, water and wind sector as those energies have zero greenhouse gas emissions.

Among the renewable energy alternatives, wind energy is the most extensively used and fastest growing renewable energy source in the world, assuming a substantial role in the global energy market and showing a compound annual growth of 21% over the last two decades [7]. It is expected to contribute to around 25% of the global renewable energy in 2040 as can be seen in Figure 2.1 [6].

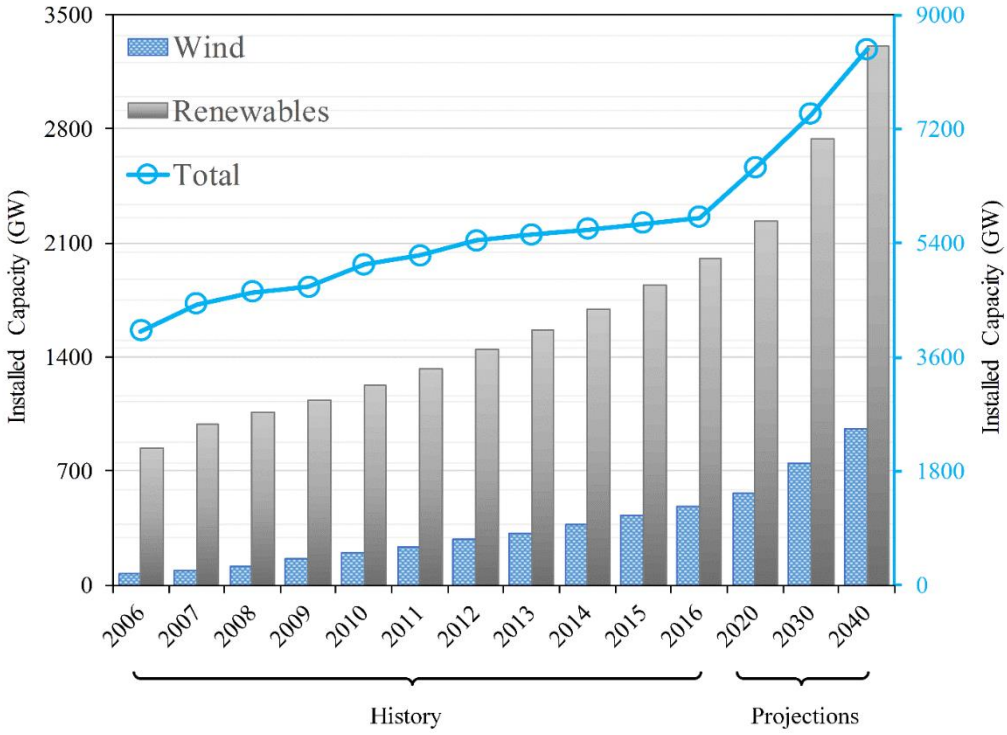

Figure 2.1: Renewable energy installed capacity by year in the world. Source: [7]

The source of wind energy is attributed to the uneven heating of the Earth's surface, as the earth is spherical different intensities of solar radiation impact the atmospheric zones close to the equator compared to the poles. This difference leads to variations in the pressure and temperature within the atmosphere which in turn causes air motion generally from the equator to the poles when viewed globally, as can be seen in Figure 2.2 [8]. In a narrower scope there are multiple mechanisms that also influence wind flow, the most significant for this work are large bodies of water which tend to have more available wind energy due to the water heat capacity causing temperature differentials in the atmosphere, as such costal and offshore areas tend to be sought after sites for the placement of large numbers of wind turbine towers [9], [10]. Another significant effect is the variation of wind speed due to height, in the region closest to the earth generally from hundreds to 1500 m depending on the topology and temperature of the region exists the atmospheric boundary layer. The wind speeds are lowest when close to the ground and increase significantly with height until the wind speed becomes constant after the atmospheric boundary layer. This greatly affects the energy yield of a particular wind turbine generator as the kinetic energy available depends directly on the wind speed.

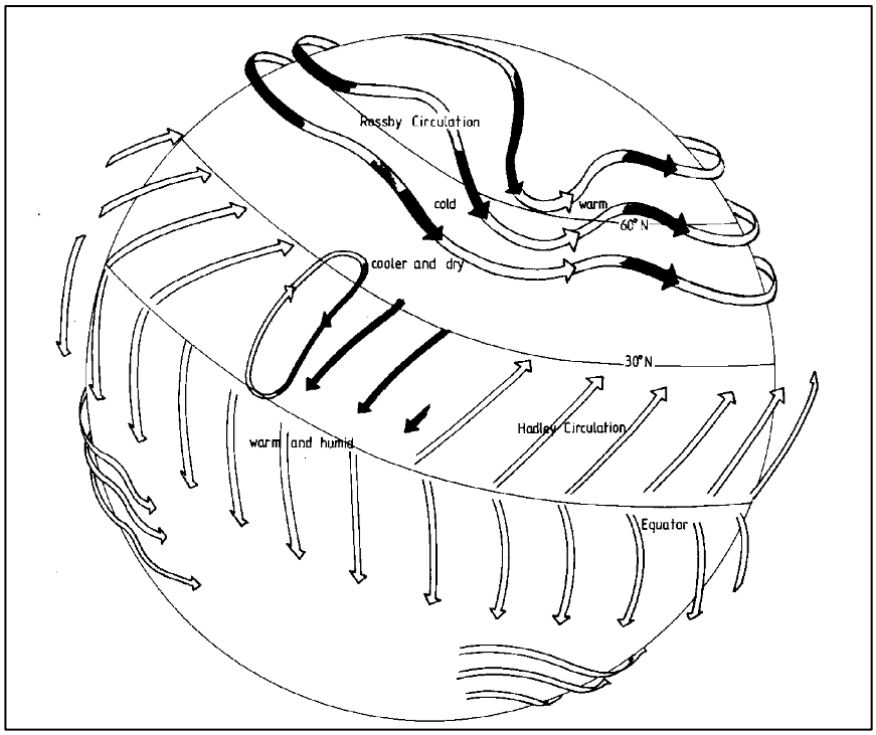

Figure 2.2: Global wind flows. Source: [8]

2.1.1. Wind Turbine Towers

Wind turbine towers are the energy converters that can convert the kinetic energy of the wind into electricity. There is a trend where the blade size has increased significantly in the past 20 years to better make use of the kinetic energy available in the air due to significant developments in composite material technology, and wind turbine tower height has also been increasing with further developments in technology [11].

Given the essential role of wind turbines in the processing of wind energy, it is imperative to provide a comprehensive overview of the key components that compose a wind turbine as well as the influence exerted by each constituent on energy production and their associated costs. Figure 2.3 illustrates the main components of a wind turbine, which can be categorized into the following segments: (i) the blades and the rotor, responsible for the initial conversion of wind energy into kinetic energy; (ii) the nacelle, which typically houses the gearbox, generator, and brakes, responsible for the subsequent conversion of kinetic energy into electricity; and, finally, (iii) the tower, responsible for providing optimal height and structural support to the other systems [12].

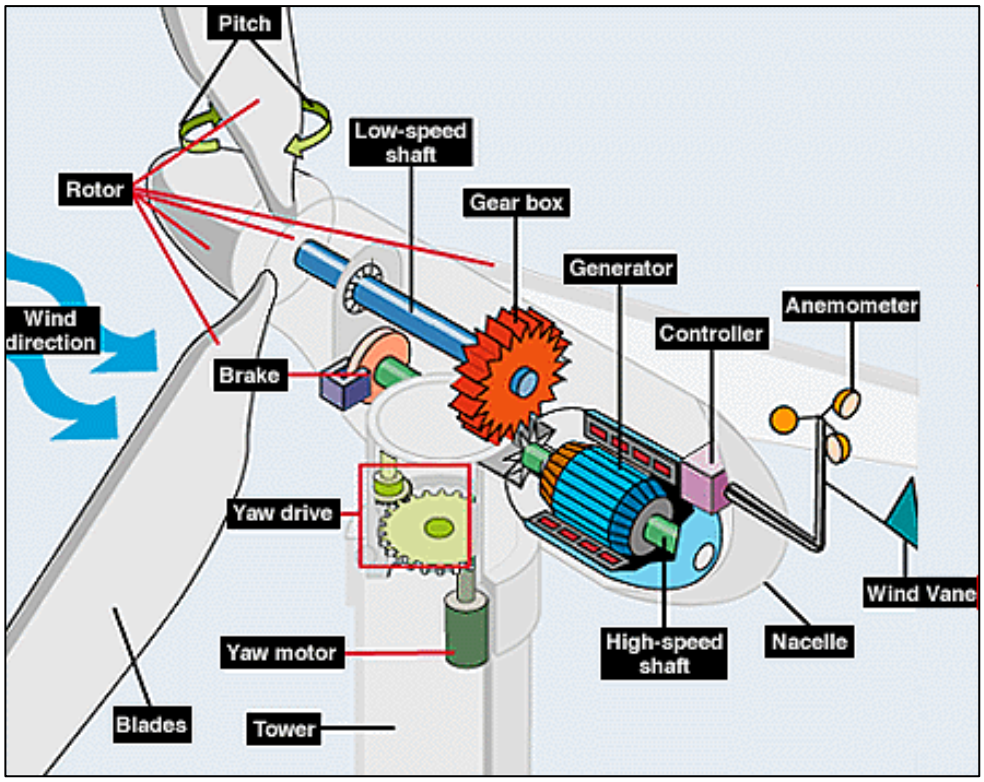

Figure 2.3: Main wind turbine components. Source: [13]

As previously mentioned, the tower is the main structural component responsible for the support of the remaining systems of the wind turbine. The height of the tower is significant as the energy yield of the wind turbine depends on the height of the wind turbine blades. In general, the higher the tower the more energy a single wind turbine produces; however, higher towers require higher costs to satisfy the strength and serviceability requirements imposed by the design codes [12].

Historically the first energy conversion systems used were simple mechanisms, the blades were connected to gearbox via a low-speed shaft, the gearbox was then connected to an induction generator via a high-speed shaft. There are multiple more modern arrangements of gearbox, generator and brakes in use and available in the literature [11], the different arrangements of mechanisms however are not significant in the scope of this work and will not be discussed in detail.

Finally, the blades are one of the most important components of a wind turbine, responsible for capturing the kinetic energy present in the wind. Due to the harsh work environment that the blades are subjected to, such as exposure to extreme temperatures, humidity, rain, hail impact, snow, ice, solar radiation, lightning, salinity and sand the wind blades can suffer from varying degrees of damage during operation [14]. As the blades and their defects are the focus of this thesis a more detailed description on the general structure of the blades will be provided.

A typical blade of large wind turbines consists of outer skins supported by a single box-shaped load carrying main spar and stiffeners and an aerodynamic shell. The blades are generally made using fibre reinforced polymeric composites and generally have a sandwich construction with low density polymer foam or balsa wood cores [11], [15]. The main load withstood by the blade is divided into three main categories, namely (i) cyclic tension loads, (ii) cyclic compression loads and (iii) shear loads. A wind turbine blade consists of two faces, due to the wind flow one face is subjected to pressure (and thus to cyclic tension loads during operation) and the other is subjected to suction (or cyclic compression loads) [14]. These faces are joined by an adhesive layer and stiffened shear resisting webs linking the upper and lower parts of the blade or the wing, as can be seen in the cross section of a blade represented in Figure 2.4.

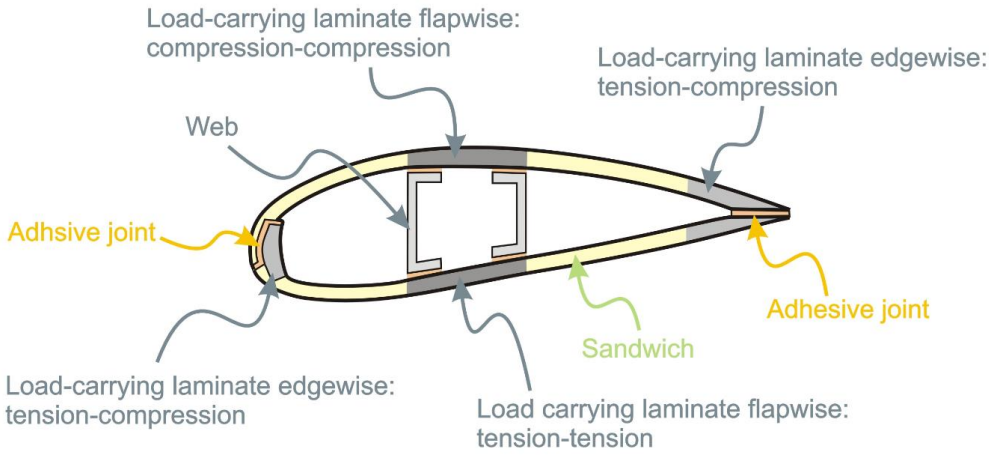

Figure 2.4: Wind blade elements. Source: [14]

As previously stated, a major trend in wind turbine development is the increase in blade size to increase the captured energy. With increased sizes, however, the weight and structural stiffness of the blade become increasingly important as to ensure that the blade does not hit the tower during operation and withstands the wind loads. In addition, as the turbines are designed to be in service for 15-20 years there is the need to verify the high-cycle fatigue behaviour of all the materials used [12].

The blades have historically been constructed by using a wide variety of materials. For electrical energy applications, before 1970 the blades were initially constructed from steel spars, with aluminium shells supported by wooden ribs, this construction method proved inefficient due to failures to cyclical damage. Afterwards wind turbines were mainly produced with composite materials composed of polymeric matrices reinforced with long fibres as they presented a much higher lifetime when compared to the aluminium blades, while maintaining an extremely attractive strength to weight ratio [14], [16].

2.2. Wind Turbine Blades

Between 1997 and 2005 in Sweden, defects in Wind Turbine Blades (WTB) were responsible for 13% of total wind turbine failures [4], [17]. While sources from the United States report a more elevated proportion, with 19.4% of the failures attributed to WTB defects [18]. WTB constitute approximately 20% of the total installation cost of a wind turbine [17]. Based on the substantial replacement costs alone, the importance of maintenance to maximize the turbine lifespan becomes evident. Furthermore, regular maintenance of the WTB is essential to guarantee the sustained productivity of wind farms as a significant WTB defect will stop the operation of the turbine in most cases for three days, while a failure can cause additional damages to the tower further increasing downtime and maintenance costs [19].

Given these circumstances, there is a need to understand and continually monitor the damage evolution in the WTB, to prevent failure of the turbines that would cause economic losses, temporary reduction of the power generation and most critically safety incidents. For the comprehension of wind blade defects a more in-depth analysis of the usual materials for WTB construction is necessary, thus an overview of composite materials, their general properties and the main manufacturing method used will be provided.

A composite material is characterized as a macroscopic combination of two or more distinct materials, which on a microscopic level possess chemically discrete phases and maintain their individual properties, with a discernible interface between them [16,17]. While often harnessed for their structural advantages, composites may also manifest distinctive electrical, thermal, tribological, or environmental characteristics. In the vast realm of engineering materials and their applications, a composite material is commonly described as a continuous matrix encompassing a secondary dispersed material. Together, these components achieve properties that surpass those of either constituent when isolated [18,19]. A complete understanding of the categorization of composite materials requires understanding of the fundamental properties of engineering materials. These materials can be classified into three distinct categories: (i) polymers, (ii) metals, and (iii) ceramics [20]. Composite materials are usually divided into three broad categories in respect to the Engineering material used as matrix constituent: (i) metal matrix composites (MMC), (ii) ceramic matrix composites (CMC) and, (iii) polymer matrix composites (PMC) [21].

Metal Matrix Composites (MMCs) are distinguished by a metal-based matrix, commonly incorporating metals such as cobalt, nickel, titanium, copper, or aluminium. These metals inherently possess attributes like high tensile strength, high stiffness, and notable ductility, complemented by high melting points and proficient electrical characteristics. Nonetheless, when compared with other material categories, they tend to have a diminished resistance to chemicals and exhibit higher densities. A prevalent industrial application of MMC is found in tungsten-carbide/cobalt composites used for cutting tools. Additionally,

their utilization in the automotive and aerospace sectors is also frequently documented [22]. MMC are not used in the wind energy sector in the production of WTB due to multiple factors, first and foremost due to the high density of the metals which results in a relatively low specific properties, while also presenting high costs and problems related to long-term cyclic damage.

Ceramic matrix composites can be segmented into two categories: (i) non-oxide and (ii) oxide-based CMC. The former encompasses carbon and SiC-based composites, while the latter includes alumina and silica-based composites. CMC display the usual ceramic material advantages such as high thermal resistances and low density. Those properties render them a compelling category of materials, especially suited for sectors like aerospace, motorsport, and other advanced applications [23]. Due to to the inherent brittleness, high costs and extreme manufacturing difficulties associated with the fabrication of large parts, this class of material is not deemed suitable to produce WTB.

Polymer matrix composites can be divided even further depending on the kind of polymer used as matrix: (i) thermoplastic, (ii) thermoset, (iii) rubber and a special subset called (iv) carbon matrix composites or carbon-carbon composites. Thermoplastic matrices have the advantage of being easily recycled due to their property of sharply reducing viscosity after the glass transition temperature is reached. Thermoset matrices reach their end state by a process defined as cross linking, that creates bonds between the multiple polymeric chains present in the material, this results in higher mechanical and thermal properties when compared to thermoplastic polymers. Due to the nature of the cross-linking reaction, after this process these thermoset polymeric matrices cannot be reshaped or easily recycled. Carbon-carbon composites represent a sophisticated class of materials comprising a carbon matrix fortified with carbon fibres. These composites exhibit mechanical properties reminiscent of ceramics. Barring the initial manufacturing phase, where phenolic resins are employed, these materials are predominantly treated akin to ceramic matrices in terms of their applications. They especially known for their outstanding mechanical properties at high temperatures, which makes them suitable for specialized applications such as aircraft brakes, rocket nozzles, and heat shields in space vehicles [24]. Composite materials can be further divided based on the form of the reinforcement material, as shown in Figure 2.5, the major categories are: particulate reinforcements, whisker reinforcements, continuous fiber and continuous sheet composites. When particulate reinforcements are uniformly distributed, they modify the matrix material's properties in an isotropic manner; a similar behaviour is observed with non-aligned whisker reinforcements. Conversely, aligned whisker reinforcements, along with continuous fibre and sheet reinforcements, enhance the matrix properties anisotropically, offering superior benefits in the specified direction. PMC share the polymer advantages of low cost, relatively low density, but they also share the

major drawbacks of low strengths, young modulus and thermal resistance, when compared with MMC and CMC. PMC are widely used in many industries such as aerospace, sports, automobile and wind energy [24], [25]. PMC are the most commonly used composites in the wind energy sector for the construction of WTB due to the ratio of the mechanical properties and the density presenting high values while also being associated with feasible costs and manufacturing for the sizes needed in the industry.

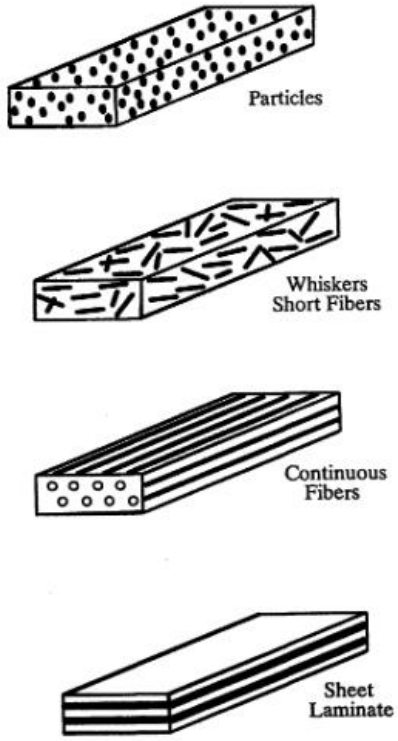

Figure 2.5: Types of composites based on the form of reinforcement. Source: [21]

As previously stated, PMC are the material of choice for rotor blades in wind turbines as these have the best balance between stiffness and density for the application. Usually thermoset polymers such as epoxy, vinylester and/or polyester are used due to the harsh environmental conditions that the WTB will be exposed to. Epoxy matrix composites are of greater interest in modern WTB due to the superior environmental resistance, surface finishing and advantages in the manufacturing process. The main reinforcement type used is long continuous fibers dispersed in the matrix. The main fibers used depend on the size of the WTB, for medium sized blades only glass fibers are used due to the lower mechanical requirements. In the more modern large blades glass fibers are used for shear and non-critical sections of the wind blade, while carbon fiber reinforcements are used for the load bearing sections. Carbon fibers are used due to the extremely high specific strength, allowing for the creation of bigger and lighter WTB, while still retaining the mechanical properties needed to withstand the loads involved. Glass fibers present

a cheaper alternative for non-critical sections of the WTB as well as a good option for the regions where shear resistance is needed due to the lower stiffness [26].

For the understanding of the defects that can appear in WTB there is a need to know and understand the manufacturing methods used to create the parts, as some of the main defects that can occur from the manufacturing. There are multiple manufacturing methods that could be used to make the large FRP parts composing the WBT; however, the commonly used ones are resin transfer moulding (RTM) and vacuum infusion process (VIP), also known as vacuum assisted resin transfer molding (VARTM). RTM is a technique that employs dry fibre laminates. Within this process, a stack of these dry fibre laminates, commonly referred to as a preform, is positioned in a mould tool. Subsequently, a secondary mould is secured over the initial one, and resin is injected at pressures exceeding atmospheric levels through one or multiple injection ports. This resin navigates the space around the dry fibre laminates via feeder pipes and a runner system. The composite is then cured in place and under pressure, as can be seen in Figure 2.6. This process, however, has an escalating cost for larger components due to the increased pressure required to ensure optimal dispersion of the resin and is used mostly for medium sized WTB [27].

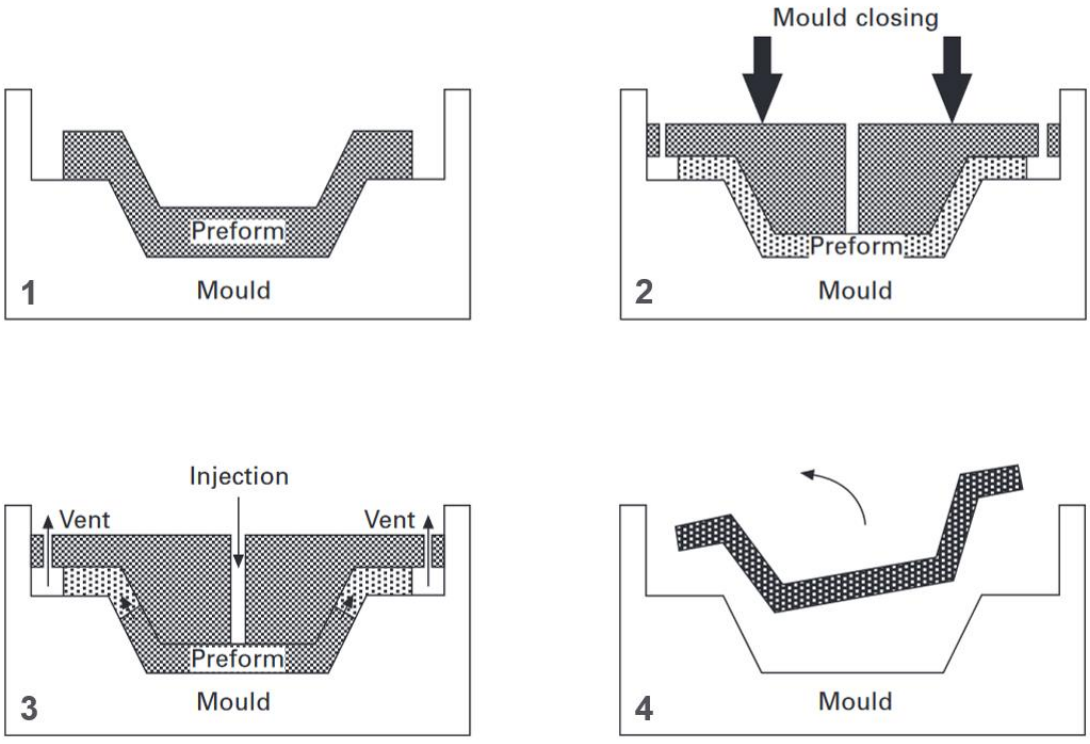

Figure 2.6: Resin transfer moulding process. Source: [27]

The VIP process is similar to the RTM process. Within VIP, dry fibre laminates or preforms are positioned in a mould. Typically, a peel-ply layer is introduced atop the preform, facilitating the facile detachment of the component from consumables and ensuring a uniform surface finish. In cases where the reinforcement possesses low in-plane permeability, distribution media may be applied over the peel-ply to optimize resin flow. Subsequently, a flexible membrane is mounted on top, and its sides are hermetically sealed to the bottom mould. By evacuating the cavity, a pressure differential is established, compacting the preform. The resin is then administered through the inlet tube, driven by the vacuum-generated pressure difference. Upon completion of resin flow, the polymer is subjected to a curing phase under vacuum conditions, as can be seen in Figure 2.7. Given that the mould undergoes minimal stress, there is a significant reduction in tooling costs, and as the pressure required does not increase with the size this manufacturing process is used to create the contemporary large-scale WTB, primarily due to the cost-efficiency in producing large components [16].

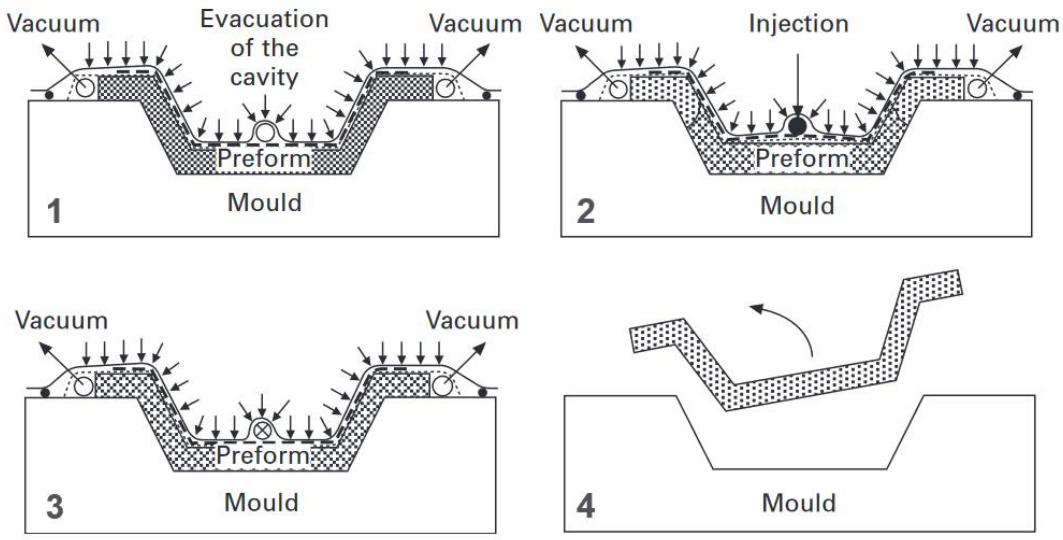

Figure 2.7: Vacuum infusion process source: [27]

2.3. Inspection Methods for WTBs

In the realm of renewable energy production, it has been noted that the cost of wind energy is substantially influenced by the maintenance expenditures related to wind towers. Delving deeper into the nuances of these costs, it becomes apparent that a significant proportion of the financial outlays can be traced back to unscheduled interruptions in turbine operations [18]. A significant fraction of these unplanned halts occurs due to the gradual exacerbation of undetected and unaddressed damages, allowing for the

damage to evolve and requiring an unscheduled stop of the operation of the turbine for the correction of the defect. This can be observed clearly in the costs associatedwith the operation and maintenance, as 57% of these costs are attributed to unplanned repairs, as documented in contemporary literature [18], [19]. Hence, inspection methodologies are pivotal to guarantee prompt damage detection, preventing the escalation that might result in turbine failure and operational downtime. Given the myriad damage mechanisms pertinent to mechanical, thermal, and environmental harms sustained by WTB, this review will address primary inspection methods employed for WTB and their respective detection capabilities along with the common defect types.

Visual inspection remains the predominant method of the assessment in the WTB inspection industry. This occurs due to the simplicity of the approach, which consists of visually inspecting the surface of the WTB to find defects. It can be executed either through direct examination by a technician, via telescopic aids, or through imagery obtained by drones [19], [26]. However, visual assessment utilizing RGB images is constrained by the large volume of data generated by drone imaging and by factors such as image quality, resolution, illumination and perspective. Defects discernible through visual inspection have been methodically classified based on depth and nature, acting as metrics for assessing damage intensity. Pertinent defect types for visual inspection of WTB are: abrasion, crack, cut, delamination, dent, gouge, hole, scratch, wrinkles, discoloration, and manufacturing anomalies and deficiencies. The most common defects that occur during in service use, accompanied by concise descriptions, are illustrated in Figure 2.8 [16].

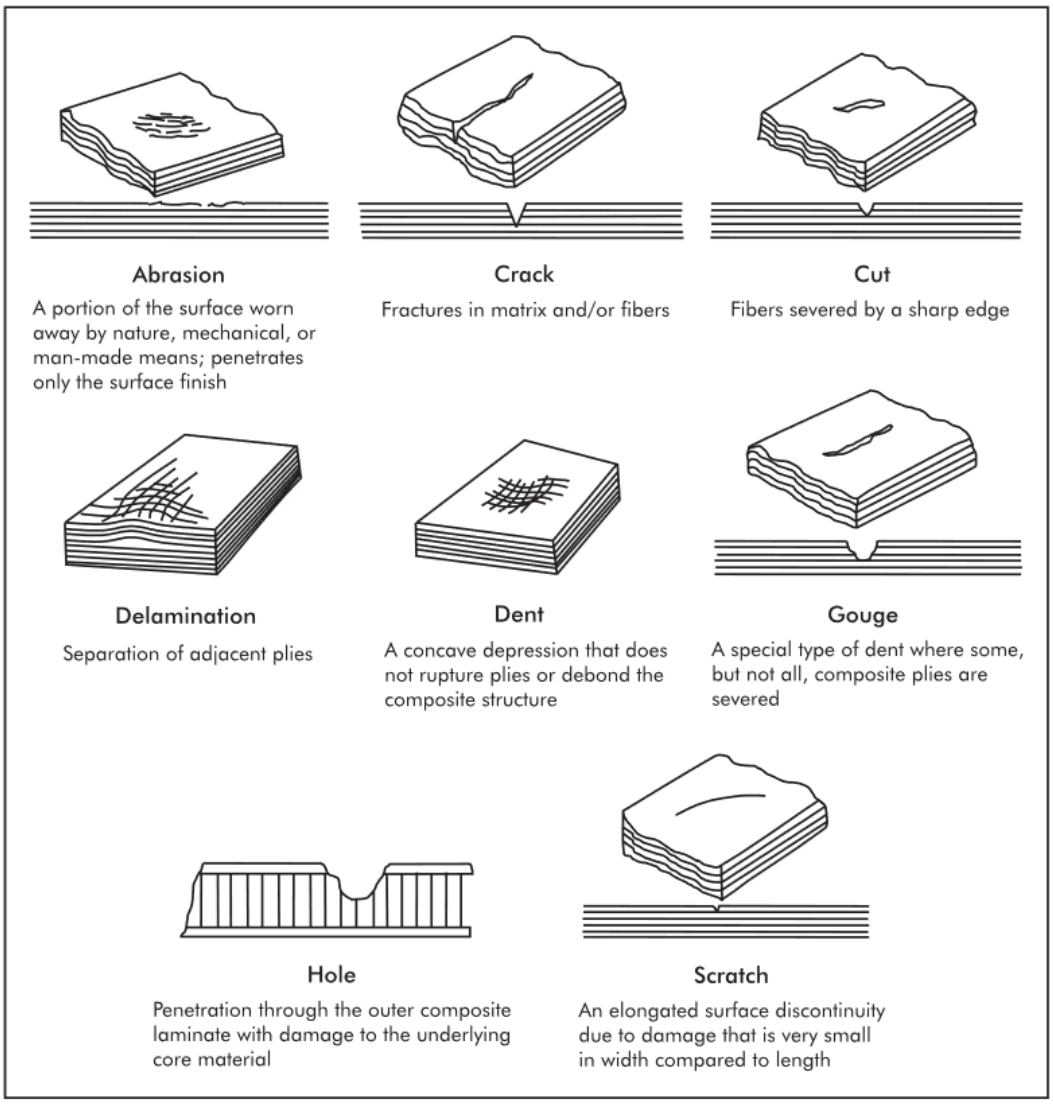

Figure 2.8: Defects of in service use detectable via visual inspection. Source: [16]

In addition to the damages arising from in-service use, certain manufacturing-induced defects are discernible via visual inspection. For composite materials, visually identifiable manufacturing anomalies typically include cracks, blisters, fibre kinks, warping, and ply overlaps or gaps. However, other manufacturing imperfections, such as delamination, porosity, voids, fibre misalignment, and variations in resin fraction, are not readily detectable through visual inspection alone [28]. The discussion of manufacturing defects is important for the general understanding of defects, however identifiable defects from the manufacturing stage should be detected before installation by the quality control, especially if identifiable via visual inspection. These defects are therefore not expected to be frequent in the inspection of working wind turbine blades. A common example is blistering, which is caused by the expansion of trapped gas within the lamina, this can occur due to chemical attack or localized heating of the matrix and is shown in Figure 2.9 .

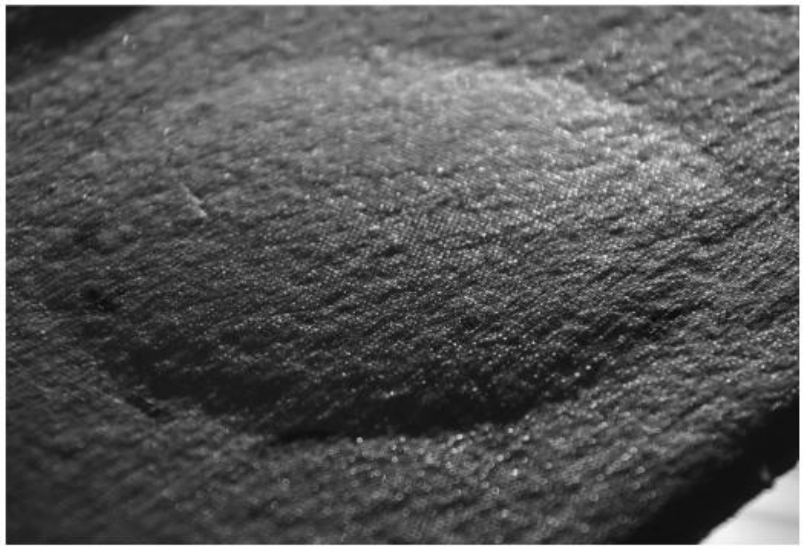

Figure 2.9: Blistering on the surface of a composite material. Source [28]

Abrasion refers to the process wherein the surface of a composite is gradually worn down through friction. It is intensified in regions with sand or saltwater and is the main cause of two of the most detectable defects in visual inspection of WTBs. The first defect is leading-edge erosion: This degradation stems from wind-driven abrasive materials consistently impacting the blade's leading edge. Over time, this wear can escalate, potentially damaging multiple slayers of the blade's shell. In visual assessments, leading-edge erosion is readily identifiable by inspecting the blade's leading edge for surface irregularities, as illustrated in Figure 2.10. The other damage related to abrasion is the damage to the coating of the shell causing exposure of the laminates below. Abrasion-induced damage can potentiate other degradation mechanisms by facilitating the ingress of moisture, introducing chemical contaminants, and enabling ultraviolet (UV) radiation exposure to the now unprotected layers.

Figure 2.10: Leading edge erosion of a WTB detectable via visual inspection. Source [29]

Cracks are a fracture in the matrix and/or the fibers of the composite, usually even small cracks are a damage that needs to be monitored in every inspection as they tend to grow in size due to the cyclic loads of the operation of the wind turbine. The location of the crack is extremely important as cracks along the adhesive material can cause much more damage to the structure of the WTB when compared to cracks of the same size on the shell. In visual inspection small cracks can be hard to detect depending on the visual inspection method used, but cracks usually present a good contrast against the white coating of the shell, as can be seen in Figure 2.11.

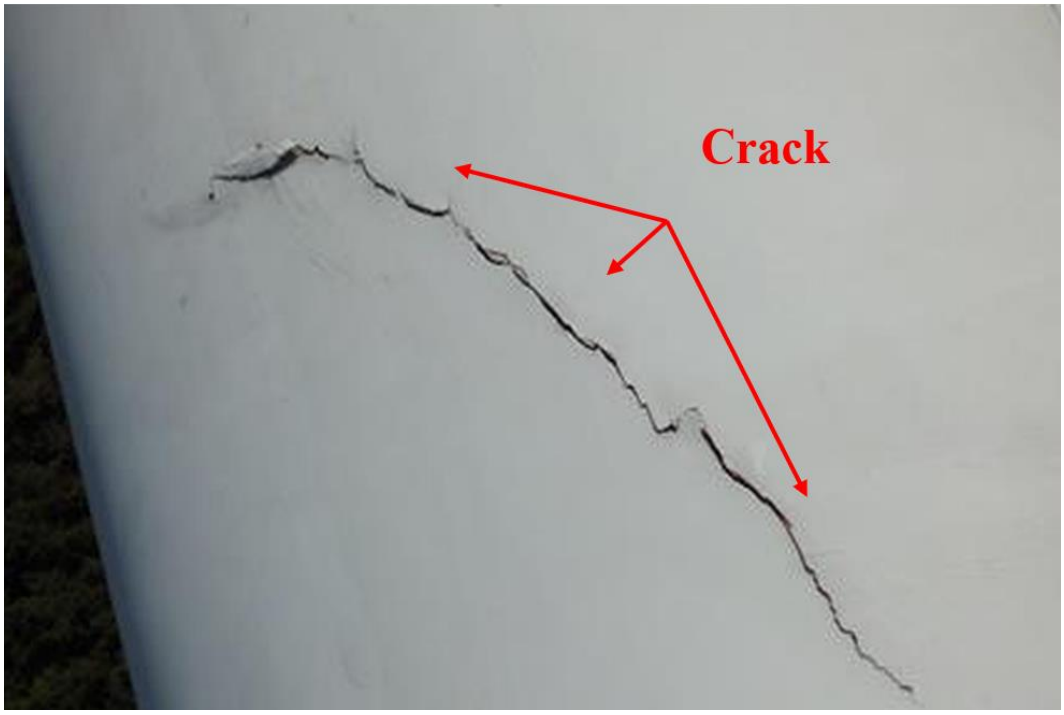

Figure 2.11: Crack in a WTB detectable via visual inspection. Source: [29]

In visual inspection dents can be mistaken for localized abrasive damage unless the damage is severe, as can be seen in Figure 2.12. Dents can be detected by visual inspection, however, the extent of the damage is impossible to measure using only visual inspection, as there may be sub superficial matrix cracking or debonding in the region around the impact. That is a reason visual inspection should not be the only inspection method used to evaluate the damage progression on a WTB and other inspection methods that identify sub-superficial damage such as shearography or thermography must be also used.

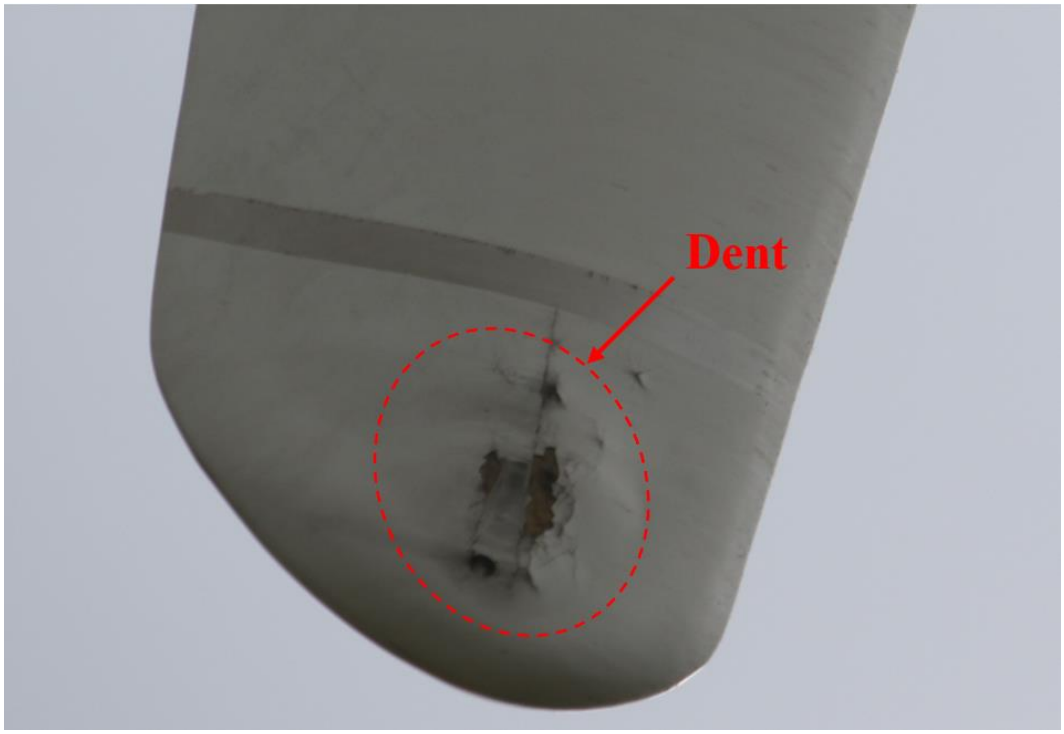

Figure 2.12: Dent in a WTB detectable via visual inspection. Source: [29]

Digital shearography is an optical measurement test that utilizes the laser speckle effect. This effect occurs when sufficiently coherent light is scattered from an optically rough surface and generates a complex granular pattern due to the interference of the light scattered from different points in the surface of the material. A small external disturbance is then applied, usually mechanical stress, and then a new granular pattern is obtained during the disturbance. The difference between the two speckle patterns allows for the analysis of the surface deformation, as can be seen in Figure 2.13, and defects such as cracks or delamination can be identified if the defect is sufficiently close to the surface of the material [3]. For the inspection of WTB digital shearography allows for the detection of subsurface damages with relatively simple equipment. However, the accuracy of the shearography analysis is affected by vibrations, which are common in WTB inspections, thus requiring careful handling or multiple inspections of the same region [30]. This solution will create large amounts of data which need to be analyzed, a recurring problem in the inspection of WTB.

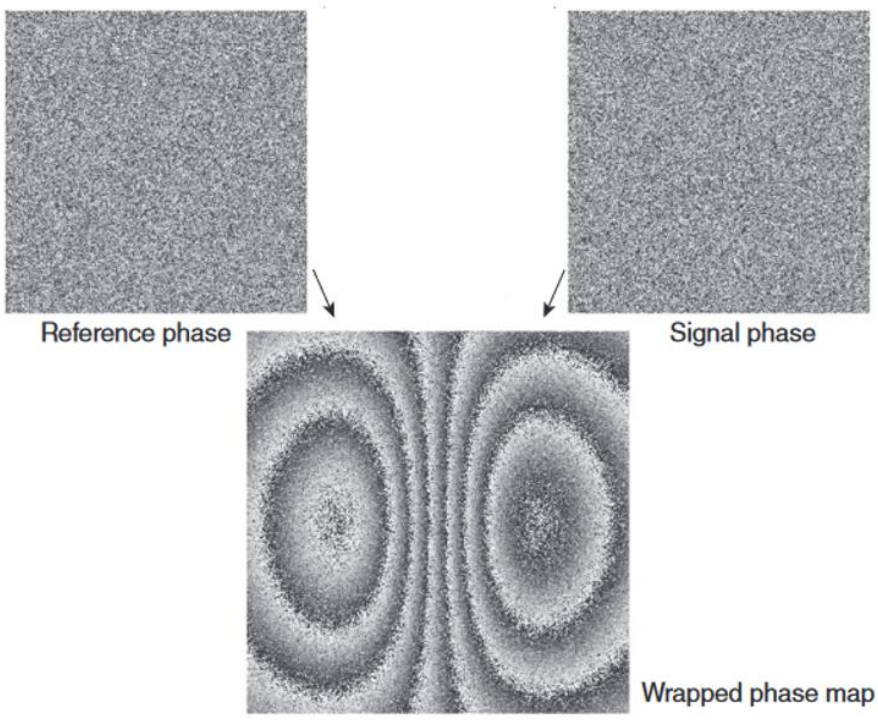

Figure 2.13: Shearography of a non-damaged material. Source: [3]

Infrared thermography is a non-destructive test (NDT) that allows for fast inspection of large surfaces. This method test consists of using infrared imaging to detect variations of temperature across the surface being tested. This test can be conducted with two different methods. The traditional thermal imaging requires thermal excitation of the sample being tested usually using lamps or heat guns, and the response of the material is then captured using the infrared camera, as seen in Figure 2.14 [31].

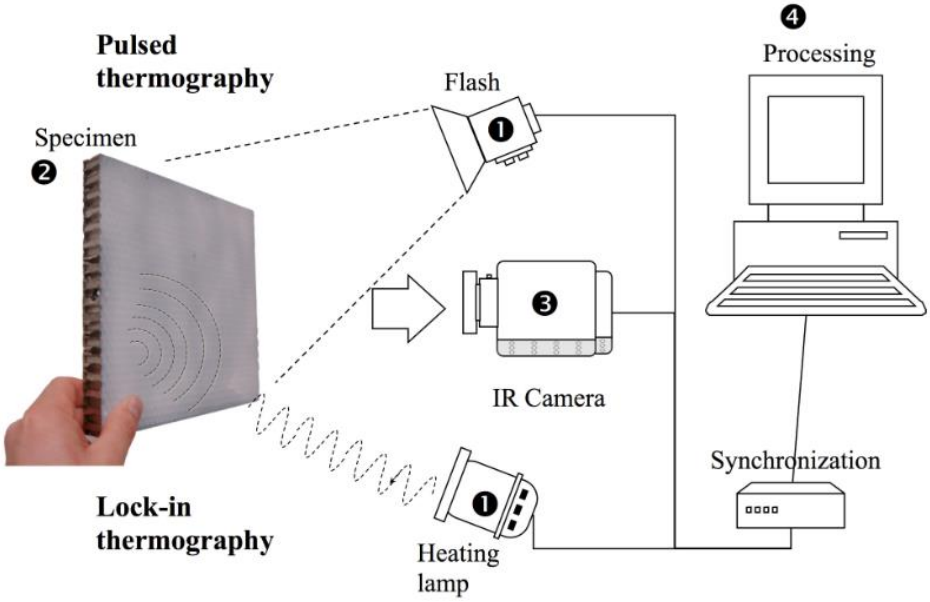

Figure 2.14: Traditional pulse thermography setup. Source: [31]

As defects modify the thermal conductivity of the material this can be used to detect sub-superficial damages in the WTB, this however, requires the interruption of the normal operation of the wind turbines for the inspection. The second method consists of the observation of the WTB while operation using a thermal camera, so that the heat generated by the growth of defects is captured and the defects observed, as can be seen in Figure 2.15 [32]. This process, however, generates great amounts of data and is not useful to find already existent defects, as only the defect growth region generates heat to be observed by the infrared camera.

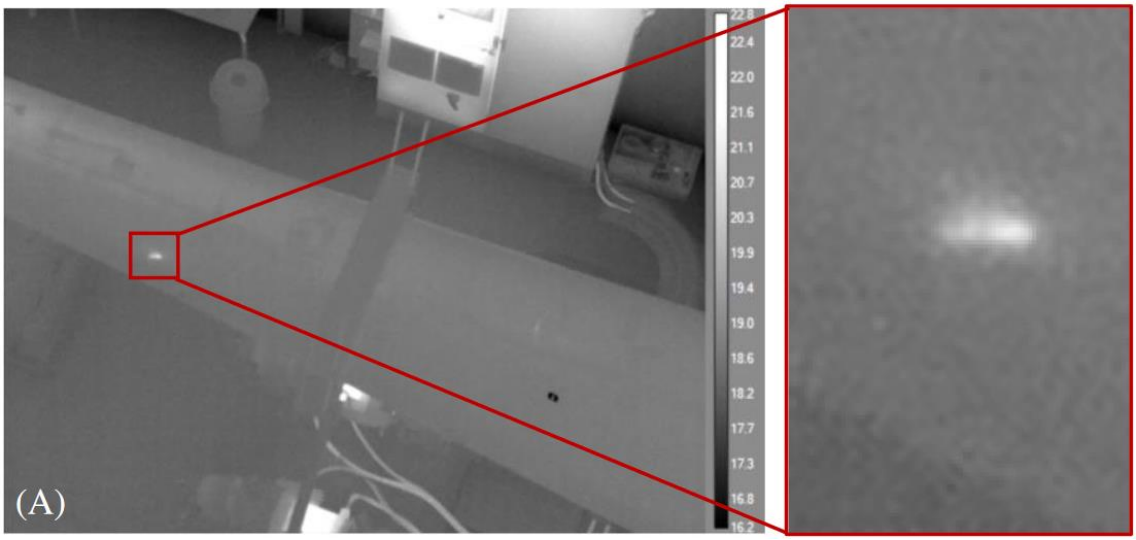

Figure 2.15: Thermography of a WTB where the crack growth tip is seen as a bright spot. Source: [32]

In WTB maintenance there is also the need to classify the damage severity, as early stages of leading-edge erosion and discoloration do not present the same damage severity as cracks, wrinkles or even more severely visible delamination or buckling of the shear webs. However, the severity categories for damage inspection are usually not made available by the inspection maintenance and repair (IMR) service provider, and as such there is no general agreement in the literature about what is the exact severity of each of the damage categories [19], [26], [33], [34]. As such one of the available damage categorization criteria in the literature will be used to exemplify the severity evaluation of different damage types and the expected impact to the lifetime of the WTB. The damages are categorized from 1 to 5 in severity where:

- Severity level 1: damages are minor variations from manufacturer specifications but within acceptable tolerance;
- Severity level 2: minor damages or defects that are outside of the manufacturer standard and outside of the tolerance and repaired damage after inspection;

- Severity level 3: minor to moderate structural damage;
- Severity level 4: significant damage or defects that have notable impact to structural capability and performance;
- Severity level 5: severe damage or defect that creates a high risk of imminent failure.

The most common damage types are assigned a severity from 1 to 5 in the damage characterization criteria selected from the literature:

- Severity level 1: Only cosmetic damage such as dirt or grease on the WTB, no damage that can be visually inspected fits this severity level;
- Severity level 2: Coating discoloration or scratches, minor wrinkles in a laminate, minor manufacturing deviations, repaired damage after inspection;
- Severity level 3: Small cracks in shells or along leading or trailing edges, small areas of under infused laminates and voids in adhesive bonds;
- Severity level 4: Prominent cracking, delamination, and shell buckling. Pronounced leading or trailing edge detachment, leaving internal components vulnerable to moisture-induced deterioration. Erosion of the leading-edge penetrating through shells, revealing the blade's internal structure;
- Severity level 5: Buckled shear webs, separation of shear webs from spar caps, significant chordwise or spanwise cracks in shells and large delamination in shells.

The objective of maintenance is to maintain the damage in the severity levels of 1 to 2, by identifying damages of severity level 3 before they can grow into severity levels 4 or 5 and significantly impact the lifespan of the WTB [35]. This classification is important for the development of damage classes for the image analysis section of the dissertation and will be further discussed in the following sections.

2.4. Machine Learning and Neural Networks

The past decade has witnessed an exponential surge in interest surrounding machine learning, fuelled by significant advancements in computational power and the development of refined machine learning tools. Machine learning, a subset of artificial intelligence (AI), involves the development of algorithms that allow computers to learn patterns and make decisions from data. Instead of relying on explicit programming for task execution, a machine learning model leverages statistical methodologies to derive insights and adapt based on exemplar data sets. As such the large amounts of data from visual inspection, shearography and thermography that were reported to present a challenge in the last chapter can instead

be turned into assets. A foundational comprehension of machine learning and neural networks is imperative to subsequently delve deeply into the intricacies of CNN [33, 34]. In general machine learning can be divided into three categories: (i) supervised learning, (ii) reinforcement learning and (iii) unsupervised learning.

The main objective in supervised learning is to create a model from labeled training data that allows for predictions on new data. The term supervised refers to the fact that all data inputs are associated with their desired output (labels), if there is a discrete number of possible outputs for the model the problem is called classification, while if there is an infinite number of possible outcomes the problem is called regression. While regression problems are familiar to most engineers and scientists due to the commonly used linear regression, in which there is a discrete set of points with labels and an infinite number of possible outcomes (the y axis), and the goal of the model is to predict an outcome (y value) given a new point (x value), an usual regression problem is demonstrated in Figure 2.16 [36], [37].

Figure 2.16: Regression problem where the output is continuous. Source: [36]

The primary aim of reinforcement learning is to determine a correspondence between input states and action outputs without initial knowledge of which actions yield optimal outcomes for given states. This process is facilitated by instituting a reward mechanism contingent on both the state and chosen action. The ultimate objective in this subsection of machine learning is to derive a policy that maps states to actions in a manner that optimizes the accumulated reward.

Finally, the main objective of unsupervised learning is to find patterns or structure within a dataset without the use of a reward function or the presence of known labels in the data. The most common uses of this category of machine learning are the use of clustering to observe relationships between data points and the use of dimensionality reduction for data compression [38].

Neural networks, a cornerstone of modern artificial intelligence and deep learning, derive their foundational concepts from our understanding of the human brain. The human brain is composed of approximately 86 billion neurons. A neuron receives signals from other neurons and if the received signal surpasses a certain threshold, the neuron activates an electrical reaction to communicate with downstream neurons. The same concept was applied to neural networks, the basic computational unit is called an artificial neuron or node. Similar to its biological counterpart, it receives input from other neurons, processes it, and produces an output. The processing often involves weighted summation of inputs followed by the application of a non-linear function, mimicking the operation mechanism in biological neurons [38]. The basic element of a neural network is a neuron, as can be seen in Figure 2.17.

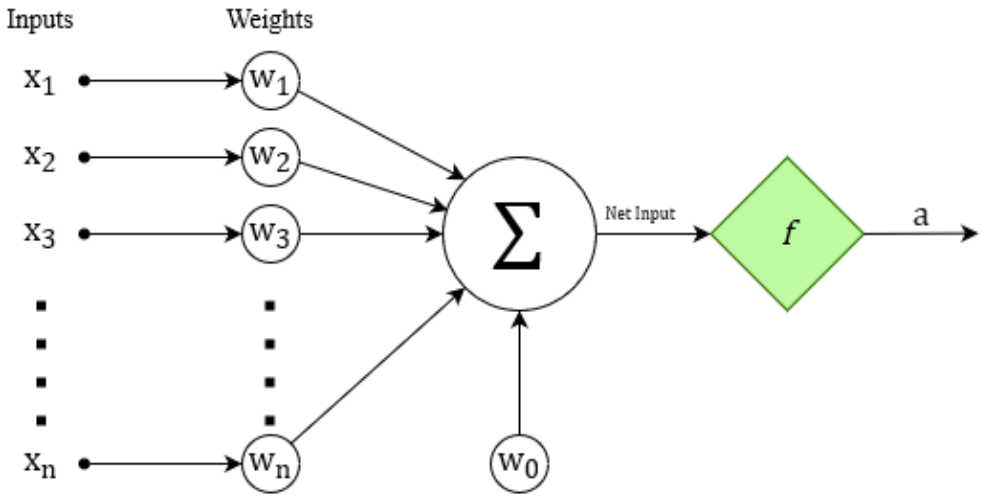

Figure 2.17: Diagram of a single neuron. Source: Author

In the neuron a set of input values are weighted and summed, and the result is used as input for an activation function that generates the non-linearity needed for the problem. Mathematically, the neuron is a nonlinear function that consists of an activation function operating on the dot product of weights and the inputs plus the bias term as can be seen in Eq.2.1 where x_j is one of the input values, w_j is one of the weights, w_0 is the bias term and f is the activation function.

$$\mathrm{a} = f\left(\sum_{j=1}^{n} x_j \cdot w_j + w_0\right) \tag{2.1}$$

In the context of CNN, another relevant aspect is the concept of a layer. A layer is a set of neurons not connected to each other. A layer is called fully connected if, as shown in Figure 2.18, the inputs to each neuron in the layer are the same. In most neural network models, all layers are fully connected.

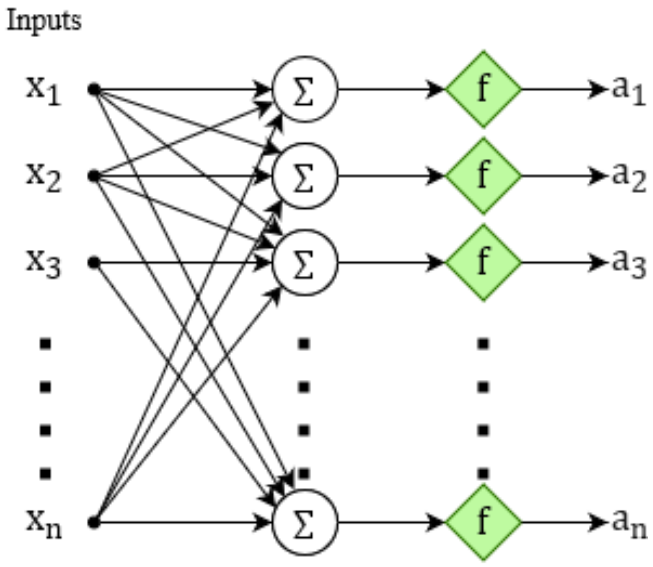

Figure 2.18: Fully connected layer. Source: Author

A neural network generally is composed of multiple layers, the most prevalent type is called the feed forward neural network, which combines layers by introducing the outputs of one layer into the inputs of the next layer, as depicted in Figure 2.19. The first layer is known as the Input layer and is tasked with introducing the data into the neural network. Typically, the number of nodes in this layer corresponds to the quantity of inputs introduced into the network.

Within a neural network, there are one or several hidden layers. These specific layers give rise to the model's non-linear behaviour. The weight values between neurons in the hidden layers are how the networks remember and adjust their results during the training. The last layer of the network is the output layer, this layer is responsible for encoding the response of the model into the final output of the model. This is done by choosing the activation function of the neurons that compose the last layer according to the expected output format.

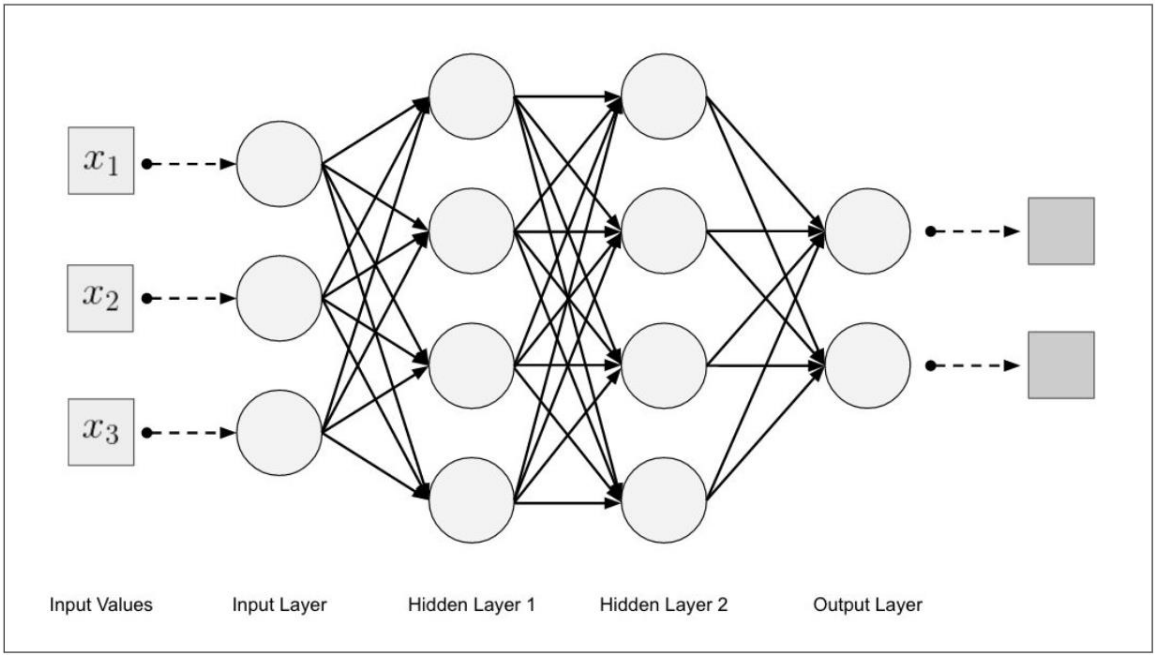

Figure 2.19: Fully connected multilayer feed-forward neural network. Source: [36]

The activation functions serve to propagate the output of the neurons of one layer forward to the next layer. They are used in the hidden layers to introduce nonlinearity into the modeling capabilities of the network. The main activation function used in the hidden layers is called the rectified linear function, it is zero for all values below zero, and linear for all positive values, as can be seen in the example provided in Figure 2.20. When used in the hidden layers this activation function solves a common problem encountered in neural networks called the vanishing gradient [37].

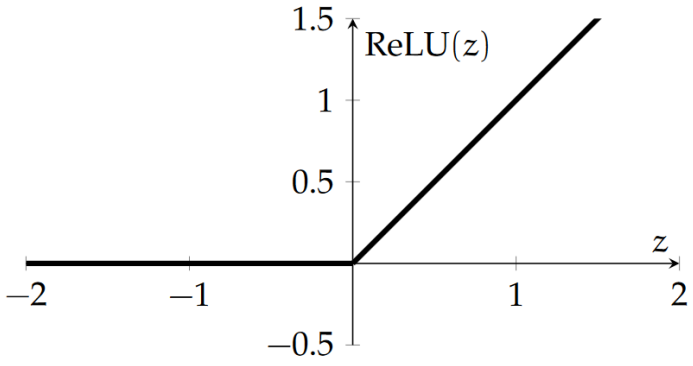

Figure 2.20: Rectified linear activation function. Source: [39]

The sigmoid activation function, also known as the logistic function, is significant as it can reduce extreme values and outliers in the data without removing them as it converts extreme points into a range between 0 and 1, as illustrated in Figure 2.21.

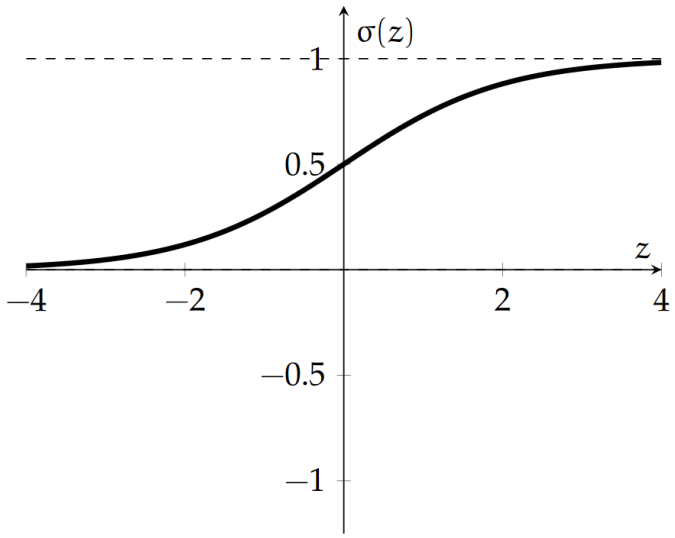

Figure 2.21: Sigmoid activation function. Source: [39]

When implemented in the output layer it can be interpreted as a probability, however, it will output independent probability for each class in the case of multiple classes, as such it is only used for binary category probability output. The sigmoid function is represented by Eq. 2.2.

$$\sigma(\mathrm{z}) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

The softmax activation function is defined by Eq. 2.3. Where z_i is the value z of the category i, and z_j is the sum of all values of z belonging to n categories. It computes the output values as a probability distribution for each class in the case of a classification problem. It is considered a generalization of the sigmoid function and in most cases is the function seen in output layers [37], [39].

$$\mathrm{softmax}(\mathrm{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.3}$$

The hyperbolic tangent is similar to the sigmoid function, it has an output range of [-1,1] as can be seen in Figure 2.22. Both tanh and sigmoid function are defined as saturated due to the fact that they have a range of possible values, as such the use of this function in the hidden layers leads to vanishing gradient problems in feed forward neural networks [36].

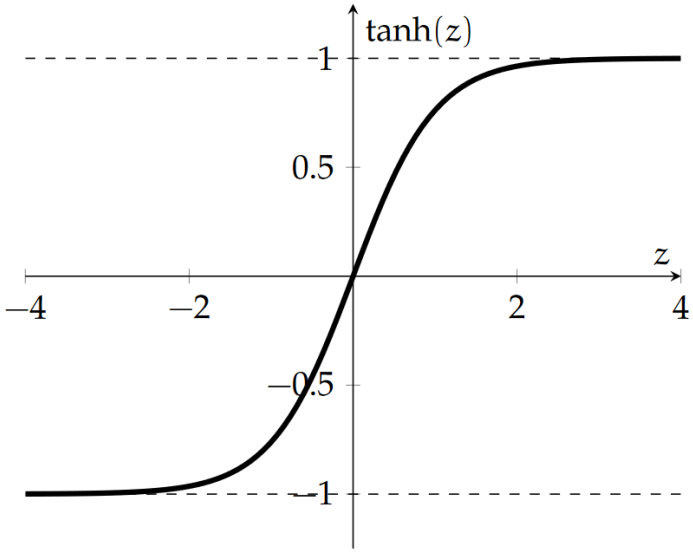

Figure 2.22: Hyperbolic tangent activation function. Source: [39]

Loss functions provide a quantitative assessment of how closely a neural network's predictions align with its training dataset. The choice of a suitable loss function hinges on several considerations: the specific problem being addressed, the optimization technique employed, and the primary goals of the training process [40]. Accuracy represents the fraction of correct predictions relative to all prediction attempts. Precision is defined as the proportion of accurately predicted positive cases against all predicted positive instances. Recall, also known as sensitivity, calculates the fraction of correctly predicted positive instances over the combination of actual positives and mistakenly predicted negatives. In the context of maintenance, it is imperative to minimize incorrect negative predictions, as this would lead to misclassifying a defect as non-damaging. Given this consideration, recall and accuracy emerge as paramount metrics, while precision assumes a secondary role. The main used loss functions for feed forward neural networks is mean squared error loss. Mean squared error loss is used when working on a model that requires an output in the form of a real value, it is widely used in feed forward neural network models and is defined by Eq. 2.4 where N is the number of data points, $\widehat{Y}_i$ is the is the predicted value of the data point i and Y_i is the actual value of the data point i.

$$L(\mathrm{W},\mathrm{b}) = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{Y}_i - Y_i\right)^2 \tag{2.4}$$

Supervised learning can be addressed as an optimization problem, in order to minimize the loss function selected for the neural network an optimization method is required. For feed-forward neural networks,

gradient descent commonly serves as the optimization technique. This method operates by determining the derivative of the loss function to ascertain the gradient. Given the multi-layered structure of the neural network, the chain rule is iteratively applied to facilitate this computation. The derivative concerning the loss is calculated at the network's final layer to deduce the gradient and subsequently modify that layer's weights. This resultant gradient is then back-propagated to determine the derivative for the preceding layer [36], [37]. This methodology is termed error back propagation and the error update rule is show in Eq.2.5 where W_{t+1} is the updated weight, W_t is the previous weight, η is the learning rate, and $\nabla J(W)$ is the gradient of the loss function with respect to the weights.

$$W_{t+1} = W_t - \eta \cdot \nabla J(W) \tag{2.5}$$

2.5. Convoluted Neural Networks

As previously mentioned, the inspection of WTB results in large volumes of data in the form of RGB images, as is the case of shearography images and thermography images. The analysis of these images can be automated by the use of convoluted neural network (CNN) models, as it is already being used in many other similar applications where there are large volumes of data in the form of images that need to be evaluated. First introduced in 1989 for handwritten digit recognition, convolutional neural networks were conceptualized based on the operational principles observed in the human cortex during object recognition processes [41]. Two salient characteristics of the human brain's visual cortex have informed machine learning adaptations. The initial observation was that many neurons in the visual cortex possess a localized receptive field, implying that they are responsive exclusively to visual stimuli within a constrained visual domain. This suggests that images are not processed as a single block, but in segments. Secondly, certain neuronal populations exhibited specialization in discerning edges and linear patterns, while others had larger receptive fields and reacted to more complex patterns that were derived from combinations of the lower-level patterns. These observations led to the idea that the higher-level neurons are based on the output of the close lower-level neurons [38], [42].

The convolutional layer stands as a pivotal element in a CNN architecture. Defined mathematically, a convolution is an operation that outlines a mechanism for amalgamating two distinct data sets. Broadly, this process ingests an input, applies a convolutional kernel or filter, and subsequently yields a feature map. Figure 2.23 graphically elucidates this procedure by showing how a filter progressively traverses the input data to generate the output. During each iteration, the kernel multiplies with the encompassed

input data values, subsequently aggregating them. This results in the derivation of a singular value on the feature map.

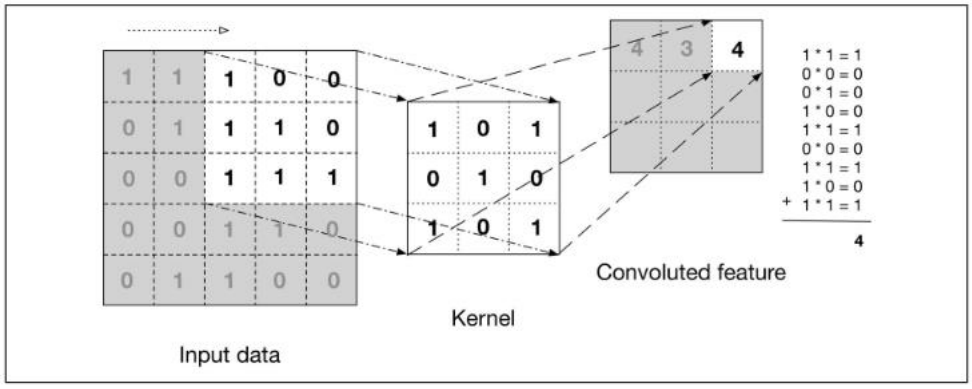

Figure 2.23: Convolution operation using a 3 × 3 filter and step of one and no padding. Source: [36]

The primary function of the convolutional layer in a CNN is to detect local patterns, such as edges, textures, and shapes. A neuron in a convolutional layer is very similar to one in a feed-forward neural network. However, the main difference is that the weights are not scalar values that multiply each pixel of the image but scalar values inside a filter that executes a convolution. The prevalent activation function employed in convolutional layers is the ReLU (rectified linear activation function), mirroring its application in feed-forward neural networks. In a convolutional layer, neurons exhibit additional hyperparameters in comparison to standard feed-forward architectures. Parameters such as kernel size, stride, and padding necessitate configuration for each convolutional layer. Specifically, the stride represents the pixel shift of the filter as it traverses the input. For instance, a stride value of 1 results in the filter moving incrementally by one pixel, whereas a stride of 2 causes a two-pixel jump per shift. Padding, on the other hand, entails the augmentation of specific rows and columns to the input to either sustain the output volume's spatial dimensions or mitigate information loss at the input space's peripheries. This is commonly achieved by zero addition. An example of padding for data conservation is a convolution operation on an input of size 5 × 5 using a 3 × 3 kernel with a stride of 1, the output if using padding is also of the size 5 × 5 and no information is lost about the borders as depicted in Figure 2.24.

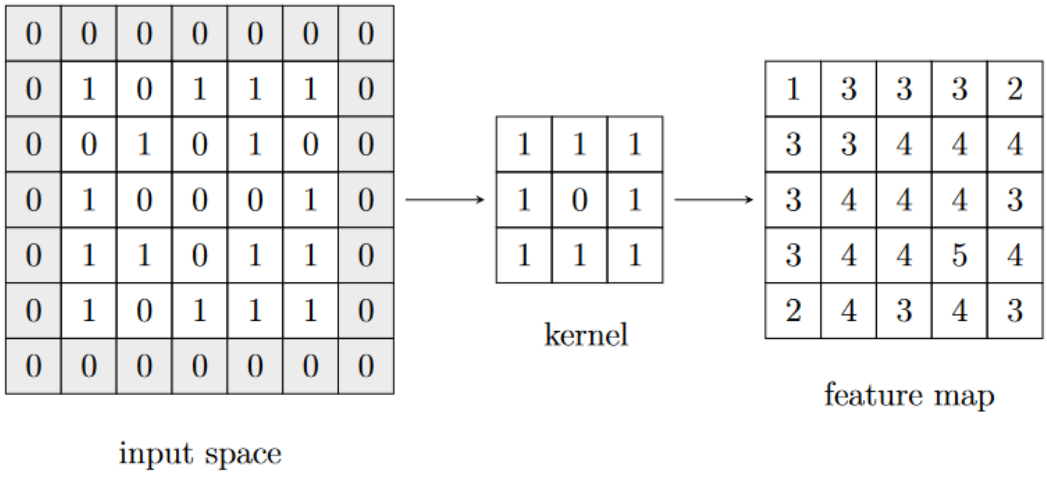

Figure 2.24: Example of a convolution operation using a 3 × 3 kernel with padding and a stride of 1. Source: [40]

In CNN architectures, pooling layers serve as another frequent component. Their primary function is to condense a specific input region into a more concise feature map segment. This is done in order to cancel out less crucial information and make outputs invariant to translations in the input space, in order to better generalize features. Common pooling techniques encompass maximum, average, and global pooling. Maximum pooling, for instance, captures the highest value within a defined region. Typically, pooling employs a stride exceeding one to diminish the dimensionality of each activation map. To illustrate, an input measuring 64 × 64 × 3, when subjected to a max pooling layer with a stride of 2, would yield a 32 × 32 × 3 feature map, as depicted in Figure 2.25 [40].

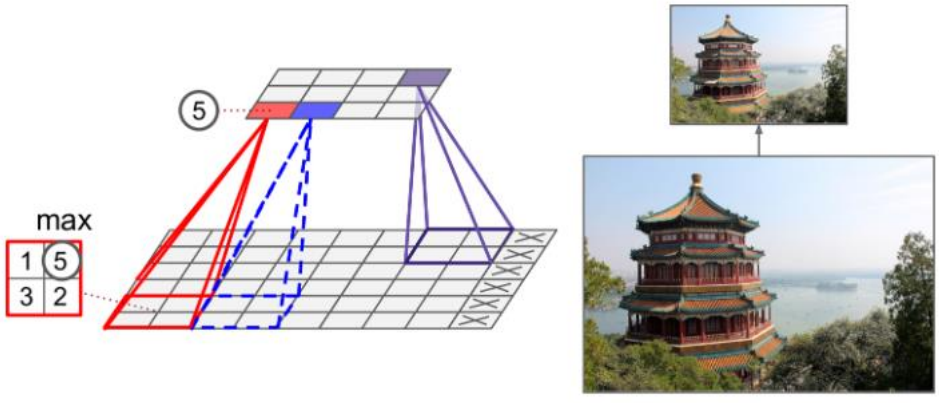

Figure 2.25: Max pooling layer with a 2x2 kernel and stride 2 [36]

A typical CNN architecture employs a sequence of layers: initial convolutional layers (each succeeded by a ReLU layer) are followed by a pooling layer. This pattern of convolutional and ReLU layers succeeded by a pooling layer recurs until the image size diminishes sufficiently. Subsequently, the data is flattened

and introduced into a fully connected feed-forward neural network. The activation function of the terminal fully connected layer must be chosen in alignment with the desired output format, a typical architecture is shown in Figure 2.26.

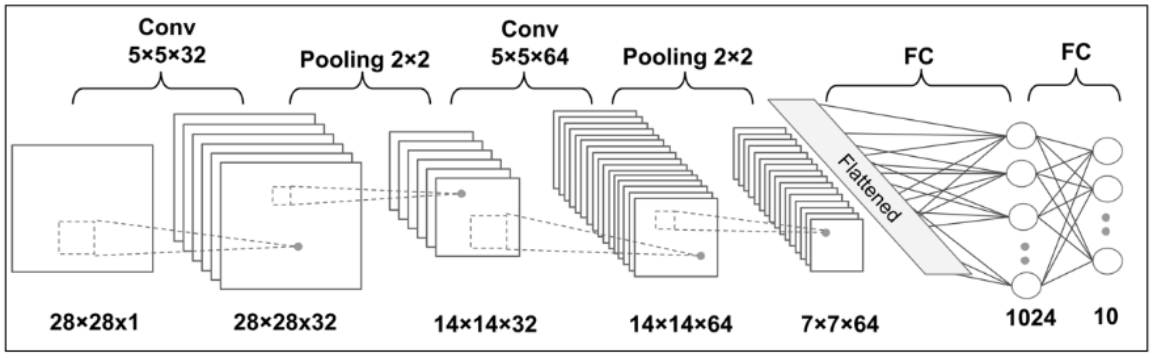

Figure 2.26: CNN architecture using 2 convoluted layers, 2 pooling layers and two fully connected layers. Source: [38]

As the general architecture of a CNN is known, there is a need to understand what are the main variables in training a convoluted neural network. As discussed in previously the training can be approached as a optimization problem, as such, the choice of optimizer and loss function is even more important on a convoluted neural network due to the large training times and processing costs involved [34], [41]. While gradient descent can be used there are multiple other optimizers developed to save processing time, costs and to allow for a greater chance of convergence to a minimal loss function. The main optimizer used in CNNs is known as Adam, which stands for adaptive moment estimation. The main equation defining Adam Eq. 2.6 resembles gradient descent, however the calculation of the terms $\widehat{m}_t$ and $\hat{v}_t$ are the improvements of this method and will be discussed in detail. In this optimizer η is the learning rate, and ε is a parameter to ensure no division by 0 occurs.

$$W_{t+1} = W_t - \eta \cdot \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.6}$$

he first new variable $\widehat{m}_t$ is defined by Eq. 2.7 where $\widehat{m}_t$ is the bias corrected moving mean of the gradients, m_{t-1} is the previous moving mean of the gradient, β_1 is a hyperparameter that controls the decay rate of this variable and $\nabla J(W)$ is the gradient of the loss function on the weights. This change in relation to simple gradient descent allows for a momentum-like property in the gradient descent due to the consideration of the previous mean gradients. This helps in dampening oscillations and ensures that

in batch process there is always movement towards the global minimum even if locally the batch does not ensure movement in that direction.

$$\widehat{m}_t = \frac{\beta_1 m_{t-1} + (1-\beta_1)\nabla J(W)}{1-\beta_1} \tag{2.7}$$

The first new variable $\hat{v}_t$ defined by Eq. 2.8 where $\hat{v}_t$ is the bias corrected moving mean of the squared gradients, v_{t-1} is the previous moving mean of the squared gradients, β_2 is a hyperparameter that controls the decay rate of this variable and $(\nabla J(W))^2$ is the squared gradient of the loss function on the weights. This moving average behaves as a stabilizer for the magnitude changes of the descent, stabilizing the values if a very high gradient is found. By using the uncentered variance (the average of squared gradients), the Adam optimizer effectively rescales the learning rate per parameter. If the squared gradient for a parameter has been large historically, the update step for that parameter will be smaller, preventing aggressive updates that could destabilize the training. Conversely, if the squared gradient for a parameter has been small, indicating that the loss landscape is relatively flat concerning that parameter, the update step will be larger, helping the optimizer to traverse flatter regions more quickly [34].

$$\hat{v}_t = \frac{\beta_2 v_{t-1} + (1-\beta_2)(\nabla J(W))^2}{1-\beta_2} \tag{2.8}$$

The use of both of these variables allows Adam to combine the benefits of both momentum-based and magnitude-based optimization techniques. The first moment facilitates a directionally consistent trajectory, ensuring momentum in the weight updates. In contrast, the second moment allows for adaptive step sizes, determined by the historical variance of gradient shifts. This dual mechanism is part of what makes Adam a popular and effective choice for training deep neural networks [43], [44].

The selection of loss functions is contingent upon understanding the output expected for the prediction. For binary outcomes, such as determining the presence or absence of a defect on a wind turbine blade, the predominant loss function employed is binary cross entropy, which is defined in Eq.2.9.

$$\text{binary CE loss} = \frac{1}{n}\sum_{i=1}^{i=n} -Y^{(i)}\log\left(-\hat{Y}^{(i)}\right) - (1-Y^{(i)})\log\left(1-\hat{Y}^{(i)}\right) \tag{2.9}$$

For multi-class outputs, like detecting and distinguishing various defects on a wind turbine blade, the selection of the loss function for CNNs hinges on the trade-off between computational efficiency and data representation. Typically, the deliberation gravitates to either categorical cross entropy or sparse categorical cross entropy. These are both a more general form of the binary cross entropy loss in Eq.2.9, with primary distinctions between both rooted in problem encoding and potential data loss. Although encoding adjustments can be readily made, they aren't typically the decisive factor in loss function selection, unless confronted with an exceedingly vast array of categories. Categorical cross entropy yields a probability distribution reflecting the likelihood of each category being accurate. In contrast, sparse categorical cross entropy solely offers the prediction of the most probable category. For scenarios with an extensive set of categories, sparse categorical cross entropy emerges as preferable due to its efficiency in reducing memory consumption during each training iteration. Both loss functions can and have been used for the detection of multi-class damage in wind turbine blades [4], [33], [43]. To assess the efficacy of a model on unobserved data, various metrics can be employed, among these the confusion matrix stands out as an essential tool. It describes the performance of a classification model, specifically a CNN, on a set of data for which the true values are known. The matrix offers a comprehensive visual representation of an algorithm's proficiency, applicable to both binary and multi-class classification scenarios. As outlined in Table 2.1 for binary classification scenarios, the matrix demonstrates the four possible outcomes of a model's predictions. These include True Positives (TP), where the model accurately identifies the positive class; True Negatives (TN), cases where the model correctly recognizes the negative class; False Positives (FP), colloquially termed "Type I error", which are instances where the model inaccurately classifies a negative class item as positive; and False Negatives (FN), commonly known as "Type II error", which are situations where the model erroneously labels a positive class item as negative [42].

Table 2.1: Confusion matrix for a binary system

Classification	**Actual Positive**	**Actual Negative**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

The main metrics used in the evaluation of CNNs are accuracy, precision, recall and the F1 score. Accuracy represents the proportion of correctly predicted instances relative to the entire dataset's

instances. It provides an overall view on the efficacy of a classification model's predictions and is defined as Eq. 2.10 where n is the number of observations. Accuracy offers a broad overview of a model's performance, but it might not always give a comprehensive perspective, particularly in datasets with class imbalances. In scenarios where one class is notably predominant, a model can achieve elevated accuracy merely by consistently predicting the dominant class. In these situations, alternative metrics such as precision or recall can offer a more nuanced evaluation of the model's effectiveness [37].

$$Accuracy = \frac{True\ Postives\ (TP) + True\ Negatives\ (TN)}{n} \tag{2.10}$$

Precision is a pivotal metric that emphasizes the accuracy of a model's positive predictions. Essentially, it measures the proportion of instances that the model correctly identified as positive out of all the instances it labelled as positive. It provides insight into the reliability of the model's positive classifications and is defined as Eq. 2.11. A measurement's accuracy pertains to how closely it aligns with the true value, while its precision reflects its consistency or repeatability. Thus, a measurement can align closely with the true value (accurate) without being consistently repeatable (precise), and vice versa. Ideally, for a measurement to be deemed valid, it should exhibit both accuracy and precision. In practice precision is used as the main metric in scenarios where the cost of a false positive is high [38].

$$Precision = \frac{True\ Postives\ (TP)}{True\ Postives\ (TP) + False\ Positives\ (FP)} \tag{2.11}$$

Recall, sometimes referred to as sensitivity or true positive rate, is a fundamental metric for classification, gauging a model's ability to accurately detect all pertinent instances. Specifically, it sheds light on the proportion of actual positive instances that the model correctly predicted as such. Recall is calculated by dividing the number of true positives by the combined count of true positives and false negatives, as illustrated in Eq. 2.12. The significance of recall amplifies in contexts where missing a positive instance (i.e., incurring a false negative) bears a substantial cost. For instance, in defect detection scenarios, recall quantifies the model's efficiency in pinpointing components with defects. Here, a false negative can inadvertently allow a defect to exacerbate, potentially reducing the lifespan of the Wind Turbine Blade (WTB). Conversely, a false positive typically prompts a manual review, ensuring the defect's verification, thereby posing lesser risks [42].

$$\text{Recall} = \frac{True\ Postives\ (TP)}{True\ Postives\ (TP) + False\ Positives\ (FP)} \tag{2.12}$$

The F1 score is a metric used in classification that conveys the balance between precision and recall. This metric is especially pertinent in scenarios with imbalanced class distributions, where striking a balance between precision and recall is pivotal. The F1 score is computed as the harmonic mean of precision and recall, delineated in Eq. 2.13. Unlike the arithmetic mean, which treats all values equally, the harmonic mean gives much more weight to low values. As a result, a model will only achieve a high F1 score if both recall and precision are high. Nonetheless, it's essential to recognize that the F1 score inherently assumes that precision and recall are equally important, which might not be the case in every application [38].

$$\text{F1 score} = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} = \frac{2\,TP}{(2\,TP + FP + FN)} \tag{2.13}$$

To thoroughly comprehend the training dynamics of a neural network, it's pivotal to delve into how the model aligns with both training and test data. Generally, it is expected that the more a model trains to fit the training data, its accuracy pertaining to that data improves. Nonetheless, this escalating accuracy on the training data doesn't necessarily translate into predicting new data, in fact there is a well reported trend in which a model where, after an extended training duration on a specific dataset, a model starts to falter in its predictions for new, unencountered data. This phenomenon is called overfitting. Conversely, underfitting occurs when the model fails to adequately capture the underlying patterns in the training data [45]. This can be attributed to either suboptimal hyperparameter choices or inadequate training epochs on the dataset. When training neural networks, if the model undergoes incessant training on identical training data, it's anticipated that its prediction error compared to time or model complexity would resemble the representation depicted in Figure 2.27.

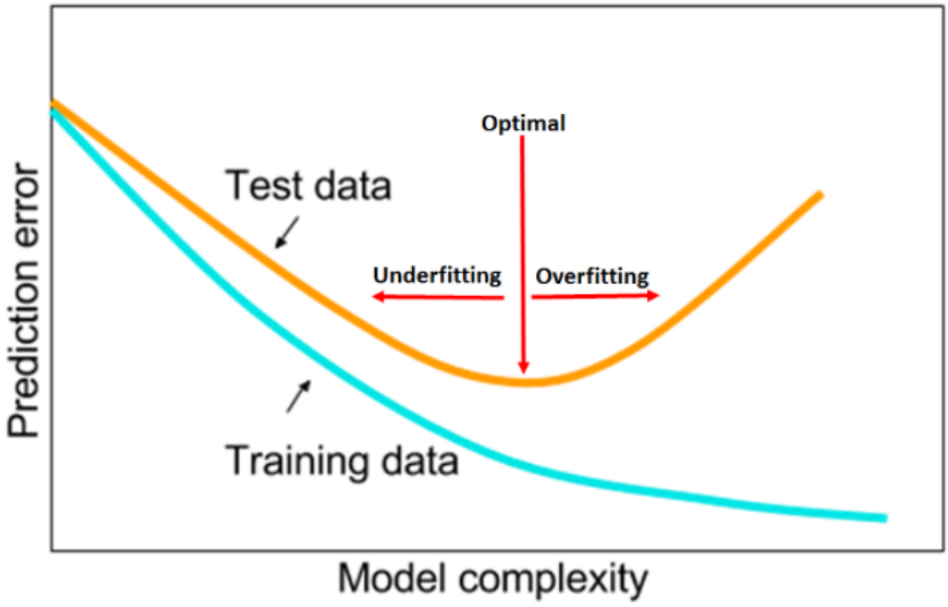

Figure 2.27: Comparison of prediction error on training and test data with increasing training, source [45]

To counteract overfitting during the training phase, various strategies, predominantly related to the model's hyperparameters, are employed. Within the realm of training convolutional neural networks, there are parameters besides the adaptable weights modified by the optimization function, that are pivotal to the training process. These parameters are called hyperparameters and can be fine-tuned to enhance the CNN's efficiency and convergence, often playing a decisive role in determining the model's final performance. These parameters govern the model's aptitude in preventing both underfitting and overfitting, the speed of which the model tries to learn the structure of the dataset, and the convergence capability of the model. The most common hyperparameters to be adjusted are the learning rate, the batch size, the activation function, momentum and regularization [45].

The learning rate is the most important hyperparameter, it is better understood when considering gradient descent, in gradient descent the learning rate is the step size taken at each iteration while moving to the minimum of the loss function, it is shown as α in Eq. 1.5. High learning rates lead to large changes in weight in the neural network during back propagation, this can lead to overshooting the minimum of the loss function leading to either oscillation that impedes convergence or divergence. Low learning rates cause small changes to the weights of the neural network during back propagation, this can lead to very slow convergence and if the training stops due to the number of training cycles being achieved can lead to underfitting. The objective with learning rates is usually to converge to a minimum of the loss function as efficiently as possible. One of the advantages of the Adam optimizer is the inherent adaptative learning

rate, in which the learning rate is adjusted for each parameter, favouring infrequently updated parameters with a larger learning rate and vice versa [37], [45].

The batch size is the number of training inputs utilized in one iteration, it is strongly dependent on the processing and memory capacity of the hardware being used, but in general batches with size bigger than 32 are not recommended as they can cause instability in the first training epochs [45]. There are three main approaches to batch size during training: Batch gradient descent, stochastic gradient descent and mini-batch gradient descent. In batch gradient descent the entire training set is processed and the means of the gradients of all training examples is used to calculate the new weights, this is the most stable form of training as updates are based on the entire dataset. However, it is often inviable in situations where the training dataset is large. In stochastic gradient descent the model weights are updated after every individual training step, this allows for very fast updates on the training weights, however it introduces a significant amount of noise in the gradient calculations and can lead to instability and failure to converge. It also does not utilize the modern hardware capability of parallel processing, leading to longer times for training. Finally mini batch gradient descent is a compromise between batch gradient descent and stochastic gradient descent, in this approach the weights are updated after every batch via the mean gradient of all the samples in a batch. In this approach the size of the batch becomes a new hyperparameter to be tuned. The most used method is mini batch gradient descent to take advantage of the parallel processing capabilities of the modern computers while minimizing noise [42], [45].

Momentum was commented when discussing the Adam optimizer previously, the main idea is that by taking into account the previous mean gradients when calculating the new weights the path of the optimizer will be smoother in the loss landscape, this can allow the optimizer to overcome small local maximums and minimums without major changes in its trajectory to the global minimum. In the Adam optimizer, this momentum behaviour is controlled by the hyperparameter β_1, typically set close to 0.9. If β_1 is set to zero, the optimizer will disregard momentum and rely solely on the most recent gradient, resembling the behaviour seen in standard gradient descent [36], [38].

The choice of activation function is critical in addressing the vanishing gradient problem. By employing the ReLU (Rectified Linear Unit) activation function for all layers, except the output layer, this issue is effectively mitigated. The consensus in the literature suggests that for most conventional data science applications, the ReLU function offers adequate performance [33].

Regularization encompasses a set of techniques to prevent overfitting used in machine learning and statistics. It generally introduces additional constraints or penalties to the optimization process, pushing the model into generalizing better for new data. The most common regularization techniques used in

convoluted neural networks are weight decay, dropout, early stopping and data augmentation. Weight decay, synonymous with L2 regularization and termed ridge regression in the context of gradient descent, incorporates an additional term to the loss function, which penalizes excessively large weights. In the general gradient descent formulation, this can be articulated as per Eq. 2.14, wherein λ serves as the hyperparameter dictating the rate of decay. Throughout the training phase, as weights undergo updates, the weight decay mechanism ensures that the weights don't grow too large unless supported by a strong gradient which helps mitigate overfitting. Weight decay is an important mechanism due to the fact that it can be used with other regularization techniques, allowing for a variety of regularization methods to be used [36].

$$W_{t+1} = W_t - \eta \cdot (\nabla J(W) + \lambda W_t) \tag{2.14}$$

Dropout is a regularization technique employed during the training phase to mitigate overfitting in neural networks, especially those with a vast number of neurons, like Convolutional Neural Networks (CNNs). Essentially, it functions by randomly deactivating (setting to zero) a fraction of neurons in a layer at each training iteration. The likelihood of any given neuron being deactivated, often around 50%, is determined by a specified hyperparameter. This strategy ensures that no single neuron's output becomes overly pivotal for the network's predictions, thereby promoting better generalization. The random "dropping out" introduces variability during training, and metaphorically, it's akin to training multiple smaller neural networks in tandem, whose outputs are consolidated during prediction. Notably, dropout is exclusively applied during training; during evaluation or inference on test data, all neurons remain active and contribute to the network's output [38], [42].

Early stopping is a technique used during model training to prevent overfitting by monitoring the model's performance on a validation dataset. As the number of training epochs increases, the model's error on the training data usually decreases. However, there can be an inflection point after which the model's performance on the validation set either plateaus or deteriorates, suggesting overfitting. At this point, early stopping terminates the training. Typically, the model's configuration corresponding to the lowest validation error observed before the rise is retained as the optimal state. This method not only ensures a model that generalizes well to unseen data but also optimizes computational resources by potentially reducing training time [38], [42].

Data augmentation encompasses the creation of novel data by implementing techniques like rotation, flips, scaling, zooming, and adjustments to brightness and contrast. The aim is to transform the original

data instances as extensively as feasible without altering their corresponding labels. To illustrate, performing a horizontal flip on a handwritten numeral "6" would inadvertently change it to a "9", thus modifying its label. Within the context of wind turbine damage detection, certain guidelines govern augmentation. Specifically, image skewing is discouraged, given the significance of retaining the blade's intrinsic shape. Moreover, any adjustment in brightness should be executed cautiously to ensure that defects remain discernible post-augmentation. Rotation involves adjusting the image orientation by a specific angle, confined within a pre-defined maximum permissible rotation. Flips, comprising horizontal or vertical inversions of the image, emerge as particularly pertinent for wind turbine blade analysis [36]. Given that most of these images are sourced via drones, it becomes imperative for the Convolutional Neural Network (CNN) to adeptly discern damages on blades, regardless of their rotational position. Scaling pertains to the proportional enlargement or reduction of image pixel count by a defined factor. When training on refined databases, scaling can be harnessed to recalibrate models for deployment on lower resolution devices. Zooming focuses on cropping the image, bringing the salient features to the forefront. Adjustments to brightness and saturation are essential for attuning the model to varying lighting conditions. While such modifications are discernible to the human eye, they furnish the CNN with invaluable data diversity. For a clearer perspective, Figure 2.28 illustrates these data augmentations, including rotation, zooming, saturation adjustments, and their combined applications.

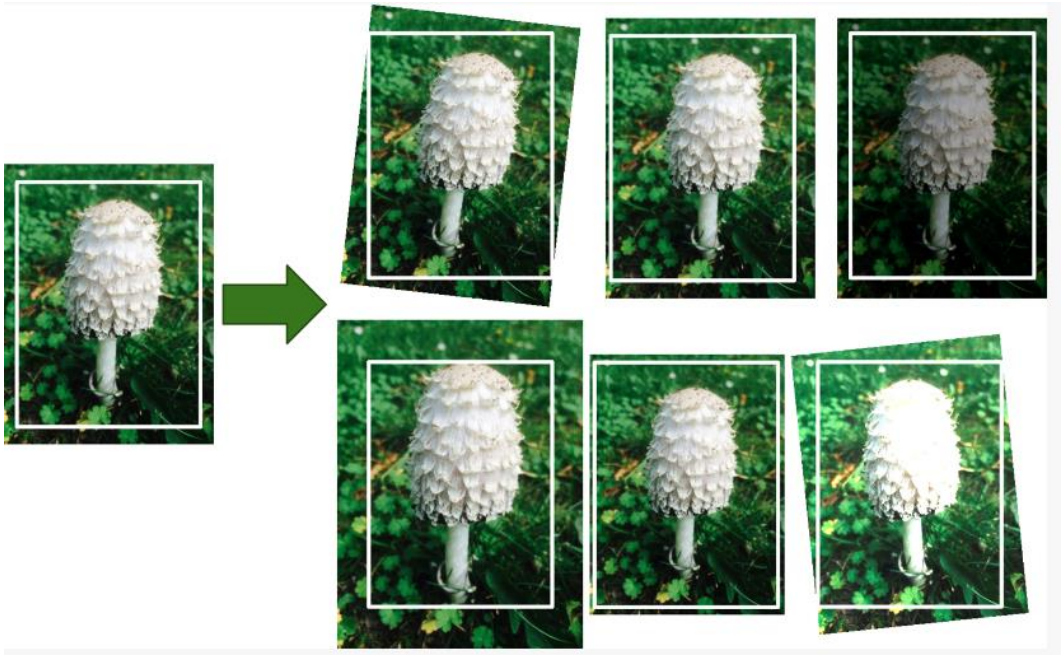

Figure 2.28: Generating new data using data augmentation for rotation, scaling and brightness changes. Source [36]

There are some techniques that allow for a faster training time or even very little computational costs on training a model, they are called transfer learning and fine tuning. There exists a plethora of pretrained models, that have undergone training with extensive data volumes, including notable architectures like ResNet, VGG, Inception, AlexNet, and others. Transfer learning consists of altering the final layer of the model and train only the final layer of the model with your dataset and proves advantageous, especially when dealing with datasets of limited volume. Conversely, fine-tuning involves retraining across all layers, leveraging the pretrained model's weights as a beneficial starting point to expedite the training trajectory [36]–[38], [42]. A prevalent strategy in the literature, albeit unnamed, seeks a compromise in the training methodology for CNNs. Initially, this approach focuses on training only the concluding fully connected layers of the CNN until the model's accuracy plateaus. Subsequently, training proceeds with a considerably reduced learning rate, targeting solely the final convolutional blocks of the model. Such a practice is tailored to acclimatize the model to the high-level patterns intrinsic to the specific dataset at hand. Typically, there's no imperative to train the model for low-level pattern detection, given that foundational patterns, such as edge and line detection, are almost universally consistent across various trained models. With an abundance of pretrained models at one's disposal, the decision regarding which model to deploy becomes critically significant. Given this, it's essential to provide an overview of the predominant models that are frequently employed in the field.

One of the first successful proposed models is the LeNet-5 architecture, it is historically significant as it was proposed by LeCun and widely used for handwritten digit recognition. As it is the simplest proposed convoluted neural network it will be discussed in detail to demonstrate how a CNN functions from input to output. It is composed of just eight layers, the first layer receives a 32 x 32 grayscale image and is composed of 6 neurons, the filter size is 5x5 the stride is one and no padding is used, therefore the output of this layer is smaller in dimensions and is 28x28x6. The next layer is an average pool layer using a filter of 2x2 and a stride of six once again with no padding, produces an output of 14 x14x6. The next layer is a convolutional layer where 16 filters of size 5x5 and stride one are used, it is important to note that unlike most modern CNNs that are fully connected this architecture does not connect all neurons of this layer to every input. This was done in order to enable the architecture to run on the available computational power of the time. The output of this layer, 10x10x16 is then passed to the next layer another average pool layer identical to the first that reduces the output to 5x5x16. The fifth layer is a fully connected layer with 120 neurons that connects to all the 400 pixels in the input, after this layer the architecture resembles a usual feed forward neural network. The sixth layer has 84 neurons and is fully connected to the previous, and the final and output layer is a softmax activation layer with 10 neurons.

This architecture demonstrates the usual process for a CNN, an image enters the architecture with a large size and is constantly convoluted and pooled until its dimensions are reduced, which are then passed to a fully connected section of the architecture which translates that input into the desired outputs [36].

AlexNet was the next significant improvement in CNN architecture, it is similar to LeNet but significantly larger and deeper, and the first to stack convolutional layers. It works with 227 x 227 sized images. It is important to note that from AlexNet forwards all architectures use the ReLU activation function for all the layers except for the output, as it prevents problems in the weight calculation during back-propagation. The full architecture of this network is shown in Table 2.2. As can be seen the first layers follow similar ideas to LeNet but with bigger filters and stride to better analyse the much larger input images [37].

Table 2.2: AlexNet architecture. Source [34]

Layer	Type	Size of the output	Filter size	Stride	Activation
1	Convolution	55 x 55	11 x 11	4	ReLU
2	Max Pooling	27 x 27	3 x 3	2	-
3	Convolution	27 x 27	5 x 5	1	ReLU
4	Max Pooling	13 x 13	3 x 3	2	-
5	Convolution	13 x 13	3 x 3	1	ReLU
6	Convolution	13 x 13	3 x 3	1	ReLU
7	Convolution	13 x 13	3 x 3	1	ReLU
8	Fully connected	4096	-	-	ReLU
9	Fully connected	4096	-	-	ReLU
Output	Fully connected	1000	-	-	softmax

Resnet is the winner of the ILSVRC 2015 challenge, this architecture is an extremely deep CNN composed of 152 layers. This depth is possible without overfitting due to the use of skip connections, where the input of layer is also added to the output of a layer located deeper in the architecture. ResNet is composed of the usual convolution and pool layers at the start similarly to both LeNet and AlexNet and then it has an extremely deep stack of residual units [38]. Each residual unit is composed of two convolutional layers, with batch normalization and ReLU activation, using 3 x 3 kernels and preserving spatial dimensions with padding as can be seen in Figure 2.29.

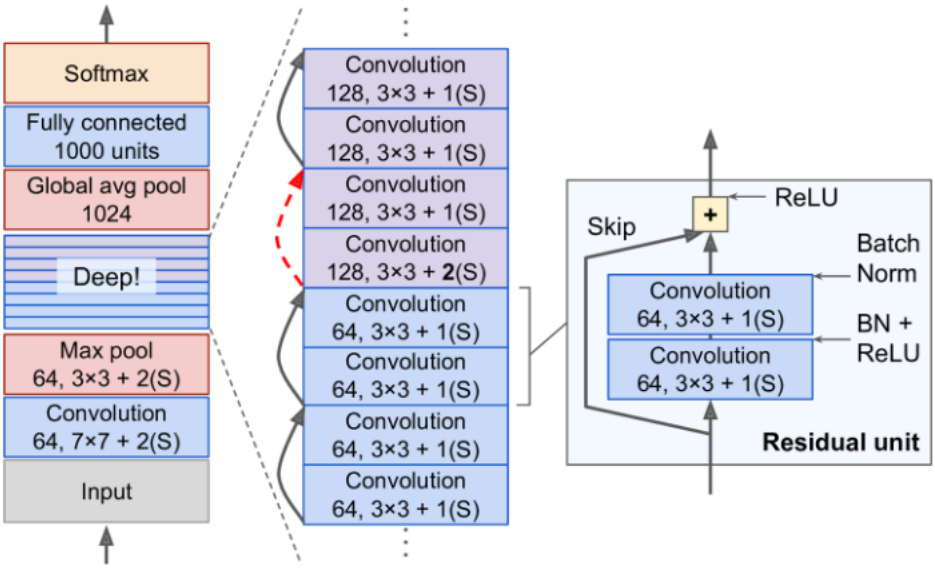

Figure 2.29: Architecture of the ResNet model source [38]

Consensus in the literature suggests that employing CNNs for damage detection in drone-captured images of WTBs is viable, with fine-tuning emerging as a favoured strategy to achieve commendable accuracy [4], [30], [32]–[34]. However, a unified agreement on the optimal model for fine-tuning WTB defect detection remains elusive. Among the contenders, ResNet50 stands out as a relatively straightforward yet efficacious model.

2.6. Summary

Wind Energy is a fast-growing sector due to international demands for more energy due to economic and populational growth and the reduction of carbon emissions due to signed treaties. A significant portion of expenses in the wind energy sector is due to the inspection and maintenance of wind towers, with a notable 57% of these costs attributed to unforeseen maintenance operations. Specifically, wind turbine blades account for the majority of these unplanned maintenance activities and entail substantial replacement costs. Consequently, there is a need to inspect and repair the WTB before non-critical damage evolves and forces a break in operations. This inspection can be done via drone imagery, however large amounts of data are generated. To process this volume of data machine learning is recommended, of the possible machine learning methodologies, convoluted neural networks are the most suitable given their adeptness at processing image data and their capability to autonomously identify defects.

A CNN needs four basic parts to be able to be trained, the base architecture, the loss function, the optimizer and the training technique. There are multiple base architectures reported in the literature, out

of them ResNet50 stands out due to the simplicity of the architecture along with a robust prediction capability. The loss functions are tailored to the form of the problem, in the context of this work they can be divided into classifying different damage categories from image data and classifying the presence of damage in the images, or multi-class and binary categorization problems. The optimizer adjusts the parameters of the model to better fit the data, usually the optimizer used in the literature is Adam. Finally two training techniques were discussed, one makes use of very little training by importing a pre-trained base, while the other uses the pre-trained base as a initial step for training.

THIS PAGE WAS INTENTIONALLY LEFT BLANK

3. IMPLEMENTATION OF THE BLADE DAMAGE DETECTION MODEL

The objective of this chapter is to detail the implementation of the model for detection of damage in wind turbine blades. The chapter starts with an introduction to the general procedures and programs needed to ensure that the convoluted neural network can be successfully created and trained, as well as detail the equipment used to train the model. Then the data received from the EDP company will be described and the procedures used to label and augment the data will be discussed. Finally, the results of the data augmentation and of the analysis of the architecture of the model will be presented.

Designing a convolutional neural network (CNN) necessitates numerous critical decisions. Foremost among these is selecting the programming language for the endeavour. The Python language has carved a niche for itself in the deep learning domain, especially in the realm of CNNs, owing to its user-friendliness, comprehensive library ecosystem, and a thriving community of practitioners [36], [38], [42]. However, alternatives like R and C++ are not uncommon. C++, in particular, finds advantages when the emphasis shifts towards optimizing performance in production settings. Given Python's rich set of machine learning and convolutional neural network libraries, coupled with the author's proficiency in the language, it emerged as the chosen medium for this project.

Choosing the appropriate Python libraries is the next step in the process of constructing CNN. The frontrunners in this arena are TensorFlow and PyTorch [36], [37]. TensorFlow, a brainchild of Google Brain, is a stalwart in the deep learning community. It’s further enhanced by Keras, a high-level API that streamlines the design process of CNN. Conversely, PyTorch, a product of Facebook's artificial intelligence (AI) Research lab, has been gaining traction, especially among the research community. Although both libraries are equally potent in setting up a full-fledged CNN, the ultimate selection was influenced by graphics Processing unit (GPU) compatibility at the start of the dissertation. TensorFlow was favoured as it had compatibility with the GPU available, facilitating computation acceleration. While PyTorch addressed this compatibility challenge within weeks, there was no pressing reason to transition between libraries. Moreover, the Keras API proved invaluable, enabling rapid construction and experimentation with diverse CNN architectures.

Training machine learning models, especially convolutional neural networks, is computationally intensive, and the choice of hardware is critical. The notebook utilized for training the models in this study is equipped with 16 GB RAM, powered by an AMD Ryzen 7 6800HS CPU with 8 cores, and benefits from the robust capabilities of an NVIDIA GeForce RTX 3060 GPU. The significance of GPU acceleration

becomes particularly evident when training complex models. To illustrate this point, the same neural network architecture (ResNet50) was trained on an identical dataset (ts.flowers). It is worth noting that the dataset and architecture were primarily chosen to underscore the time disparities in training, with a deeper examination of the databases and architectures to be presented later in this chapter. Utilizing only the CPU, a single epoch for training the model was projected to consume 54 minutes and 20 seconds, as indicated in the program's output depicted in Figure 3.1. Given that models typically require anywhere from 15 to 50 epochs, training on the CPU alone could easily extend beyond 13 hours. It is also significative to mention that the loss and accuracy metrics in Figure 3.1 are not conclusive since the epoch's training was not completed.

```
Epoch 1/5
 1/86 [..............................] - ETA: 54:20 - loss: 2.7089 - accuracy: 0.2500
```

Figure 3.1: Training of the example model using the CPU. Source: Author

In stark contrast, leveraging the GPU drastically accelerates the training process, reducing the time required for one epoch to a mere 10 seconds, as demonstrated in Table 3.1. Consequently, the complete training of this specific example can be accomplished in less than a minute. Importantly, it is essential to recognize that the model employed for this instance wasn't fine-tuned or optimized for the task, serving solely as an illustrative representation of the time commitment associated with the analysis. Even under these conditions, the model managed to achieve an approximate training accuracy of 86%. Although this figure may seem modest by contemporary standards, it is noteworthy that it matches the performance of cutting-edge models from a decade ago, such as AlexNet.

Table 3.1: Training of the example model using the GPU

Epoch	Time	Test Accuracy	Validation Accuracy
1	10 seconds	0.9004	0.8438
2	7 seconds	0.9197	0.8658
3	7 seconds	0.9350	0.8419
4	7 seconds	0.9433	0.8603
5	7 seconds	0.9517	0.8603

Due to an unforeseen delay in acquiring the wind turbine blade data essential for model training, preliminary steps were taken to train and compare two architectures on available unrelated datasets. This

exercise aimed to discern the varied outcomes stemming from employing transfer learning and fine-tuning techniques, as well as contrasting results across different model architectures. Section 3.2 details the dataset used for the initial comparisons as well as the dataset used for the WTB defect detection training, the steps taken for labelling the dataset and the data augmentation procedures used in the dataset. Section 3.3 delves deeper into the specific details of the chosen architectures and the associated methodologies for both fine-tuning and transfer learning. As well as a comprehensive analysis comparing the models and contrasting the approaches of fine tuning and transfer learning on a dataset. An essential caveat to consider is that while the observations drawn from a particular dataset can shed light on the potential performance of various CNN architectures, and on the efficacy of fine tuning versus transfer learning, one cannot make sweeping generalizations. This is because the specifics of the dataset in use such as number of data points and image quality, coupled with the anticipated output categories, significantly affect how a model will train and perform.

3.1. Data

Due to unforeseen delays in acquiring the specialized data for wind turbine blade (WTB) defect detection, an alternative approach was necessitated. In order to validate the proposed architecture and the training methodologies, a readily available database was employed for preliminary testing. The TensorFlow flowers database was the chosen dataset, primarily because it is extensive and requires the differentiation of visually similar objects across a limited set of categories. This dataset encompasses 3,670 RGB images, each measuring 256 × 256 × 3 pixels, which are distributed among five distinct flower classifications. Given the richness of this dataset, there was no need to resort to data augmentation. A visual sample from this dataset, presenting various flower images and their corresponding labels, is depicted in Figure 3.2.

Figure 3.2: Random images from the TensorFlow flowers dataset. Source [46]

Upon receiving the dataset pertaining to wind turbine blade (WTB) damage from EDP, it became evident that the data was insufficient for the comprehensive training of a convolutional neural network. For context, the literature cites that the smallest dataset employed for CNN-based WTB defect detection comprised 1,700 images before data augmentation [33]. Other datasets mentioned in research circles boast figures like 3,000 images [4] and a sizable 71,000 images [43]. Understandably, the more extensive datasets yield superior results; the largest dataset reported a commendable accuracy of nearly 91% on test data. In contrast, due to several constraints out of control of EDP, the dataset provided contained 20 images, predominantly showcasing leading edge erosion damage. To bolster the dataset for training purposes, drone imagery from the publicly accessible Nordtank dataset was incorporated [29].

Upon collating the data, a manual labelling process was undertaken to classify the types of defects found in the images. In total, the amalgamated dataset boasted 100 images: 50 portrayed an undamaged WTB, while the remaining 50 depicted various defects. Leading edge erosion was the most prevalent with 22 images, trailed by cracks (12 images), coating damage (8 images), UV exposure (4 images), and impact damage (4 images). The limitation posed by the sparse dataset necessitated a re-evaluation of the objectives. A multi-category classification, given the volume of data, was determined to be unfeasible. Thus, the scope was shifted to a binary classification task, focusing on detecting the presence or absence

of a defect, regardless of its specific type. The binary classification, in contrast to a multi-category one, allows for a more streamlined utilization of data, especially when data is limited. By consolidating all defect types into a singular 'damaged' category, this approach effectively maximizes the utility of available data. An example of the images in the dataset after the reclassification is shown in Figure 3.3.

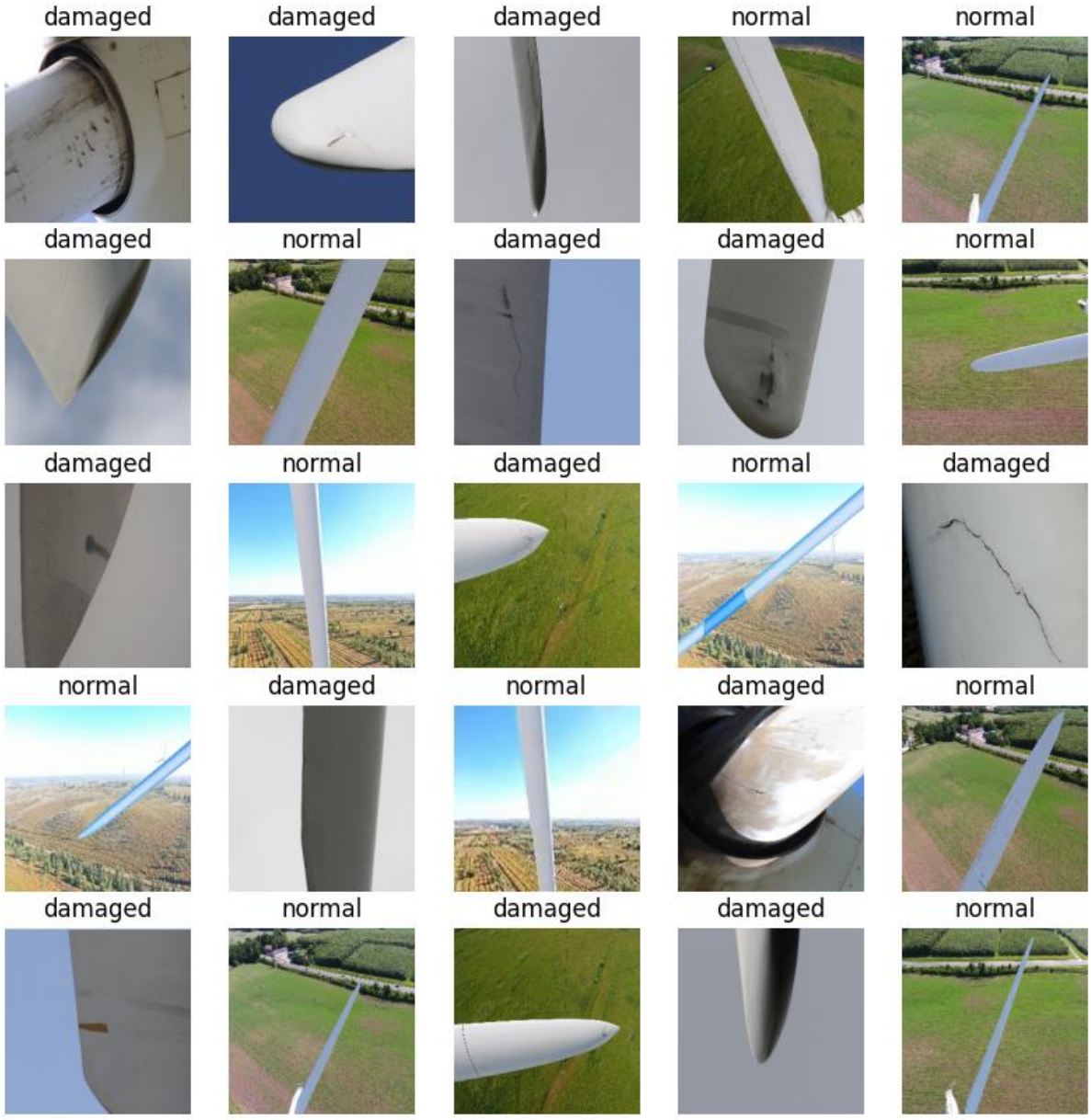

Figure 3.3: Example of images on the dataset along with the new classification. Source: [28]

To further enhance the dataset, data augmentation techniques were employed. Using a Python script, each image from the dataset underwent several transformations: vertical and horizontal mirroring, random rotations capped at 5 degrees, and variations in brightness and saturation ranging from 60% to 140% of the original image's value. These augmentative procedures yielded four distinct images for every original data point, thereby quadrupling the dataset size and enhancing the potential for model training.

An example of an original image of the database along with the four data augmented images generated is shown in Figure 3.4.

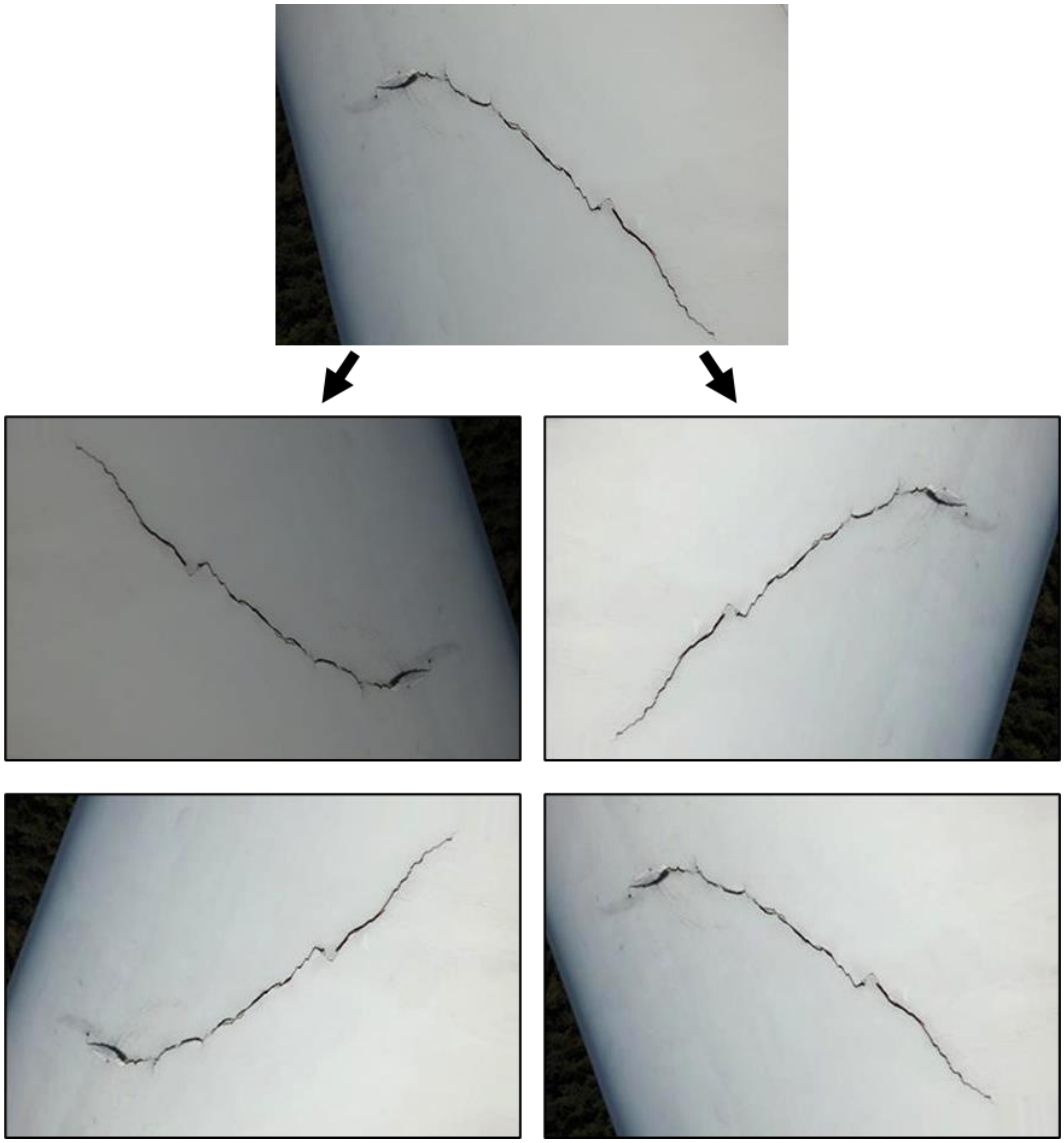

Figure 3.4: Data augmentation on one of the images of the dataset. Source: [28]

3.2. Convoluted neural network architecture

As discussed in the state of the art, the ResNet architecture is one of the most used architectures used to this day in the object detection and image categorization fields. This architecture is divided into the number of residual networks it uses, or in other words the depth of the model, ResNet architectures come in various depths, including ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. The number denotes how many layers with weights the model has in its architecture. This includes convolutional and fully connected layers. In general, the use of ResNet50 is the recommended as it presents a good balance

between training speed, number of parameters and prediction capabilities, it is also the architecture used in some of the WTB defect detection CNN reported in the literature [33].

Fine-tuning and transfer learning are cornerstone methodologies in the domain of convolutional neural networks when the objective is not to devise a novel architecture from scratch but to utilize established, pre-trained models to achieve efficient image classification or object detection. To comprehensively grasp the nuances and implications of these techniques, an experimental approach can be insightful. The ResNet50 architecture was selected due to its depth and established performance on tasks like image classification. Transfer learning and fine-tuning strategies were implemented separately on TensorFlow's flowers database and subsequently compared on their predictive capabilities. This experiment aimed to offer tangible evidence of how these methodologies impact the efficacy of the model. While both approaches have their merits and are anchored in leveraging the strengths of pre-trained models, understanding their direct influence on model performance can inform the choices made for the application in the use of damage detection.

Implementing transfer learning with TensorFlow coupled with the Keras API is straightforward. The preliminary step involves importing the dataset. Subsequently, this dataset undergoes a split: typically, as corroborated by numerous studies in the literature [4], [33], [36]–[38], [42], [44], 75% of the dataset is earmarked for training, while the remaining 25% is set aside for validation. Once the dataset is appropriately segregated, the next step in the transfer learning process is the importing of a pre-trained model. Using the Keras API streamlines the setup of the ResNet50 architecture. It requires three primary inputs: the weights intended for use, a boolean to decide whether the model's final layer (or "top") should be incorporated, and the anticipated input shape for the network. By choosing 'ImageNet' for the weights, we capitalize on the extensive features and patterns the model learned from this colossal dataset, which comprises over 150,000 images spread across a thousand distinct categories. The input shape parameter is adjusted to the dimensions of the images present in the flower database. However, the task of classifying flower species diverges from ImageNet's extensive categorization. Thus, adjustments to the model's concluding layer are imperative. Fortunately, the Keras API once again simplifies this process, as it offers a straightforward mechanism to exclude the original output layer of the architecture. Afterwards, the addition of a fully connected layer tailored to the number of flower species in our dataset is a very simple process and ensures the model produces appropriate outputs. To ensure that only this new layer undergoes training, and the core architecture derived from ResNet50 remains untouched, the layers of the pre-trained model are designated as non-trainable.

Upon making the necessary modifications to the ResNet50 architecture, several crucial decisions must be made to ensure the effective training of the model. First, the choice of a loss function is pivotal. Given the dataset's multi-class classification nature, the 'categorical cross entropy' loss is the most fitting. Next, the selection of an optimizer is crucial in dictating how the model updates its weights in response to the observed data. For this task, the Adam optimizer is selected, the hyperparameters selected for the training are the usual values in the literature and are represented in Table 3.2 [37], [42].

Table 3.2: Hyperparameters selected for training.

Hyperparameter	**Values selected for the training**
learning rate (η)	0.001
momentum parameter (β_1)	0.9
adaptive learning rate parameter (β_2)	0.99
weight decay parameter (λ)	0

As explained in the state of the art this optimizer presents significant advantages over the usual gradient descent optimizer. Metrics are essential as they provide insight into the model's performance during training. For this project, the 'accuracy' metric is employed. This offers a straightforward measure of the proportion of correct predictions made by the model, making it a suitable choice when the primary goal is a basic evaluation. Finally, with the configuration in place, the model training commences using the training dataset. Simultaneously, its performance is monitored on the validation set. This dual approach provides insights into how well the model is learning and helps in early detection of issues like overfitting, ensuring the development of a robust and generalizable classifier.

The results for 15 epochs of training on the dataset using the are shown in Table 3.3. The training phase involves the adjustment of the model to predict the training data. This adjustment is orchestrated by modifying the weights inside the neurons using the optimizer function. Meanwhile, validation data, or data of samples not utilized during training, is used to assess the model's predictive accuracy on unseen data. As detailed in section 2.5, accuracy is an aggregate metric representing the ratio of correct predictions to the total predictions made, mathematically depicted in Eq. 2.10. It can be seen that while the accuracy on the training data goes up with each training epoch, as is expected, the accuracy on the validation data remains around 87% for all the epochs which means that the model is not learning to generalize more with the remaining training steps. No early stopping mechanism was defined due to the interest in observing if the model would avoid overfitting. The best performing model was trained on epoch 10 with 87.5% accuracy on the validation data.

Table 3.3: Results for 15 epochs of training using the transfer learning technique.

Epoch	Time	Test Accuracy	Validation Accuracy
1	11 seconds	0.9204	0.8640
2	7 seconds	0.9350	0.8640
3	7 seconds	0.9331	0.8695
4	7 seconds	0.9386	0.8732
5	7 seconds	0.9397	0.8658
6	7 seconds	0.9353	0.8695
7	7 seconds	0.9404	0.8676
8	7 seconds	0.9324	0.8676
9	7 seconds	0.9430	0.8676
10	7 seconds	0.9469	0.8750
11	7 seconds	0.9426	0.8676
12	7 seconds	0.9509	0.8658
13	7 seconds	0.9480	0.8695
14	7 seconds	0.9444	0.8713
15	7 seconds	0.9506	0.8603

Fine-tuning closely mirrors the process of transfer learning in its initial stages but differentiates primarily in how the pre-trained model is treated. Beginning with the same dataset, the foundational ResNet50 architecture is established using Keras. The model's final layer is then removed, and a fully connected layer suited to the specific class distribution of the flower database is incorporated in its place. The main difference between fine tuning and transfer learning is that while transfer learning retains the base model's layers as fixed, fine-tuning permits all layers to undergo training and adaptation. Subsequent to configuring the architecture, the activation function, optimizer, and metrics are designated identical values to those employed in the transfer learning phase, ensuring a consistent evaluation basis between the two techniques. Due to the more comprehensive nature of fine-tuning, which adjusts the weights across the entire model, it is anticipated to require a longer training duration compared to transfer learning. Accordingly, a greater number of epochs, 30 in this case, are allocated for training. To provide a more intuitive grasp of the model's performance trajectory over the epochs, results from the fine-tuning process are visualized graphically in Figure 3.5 as opposed to a tabular format. This facilitates a clearer understanding of the model's evolution and highlights any notable trends or potential issues. The best performing model was trained on epoch 29 with 87.5 accuracy on the validation data.

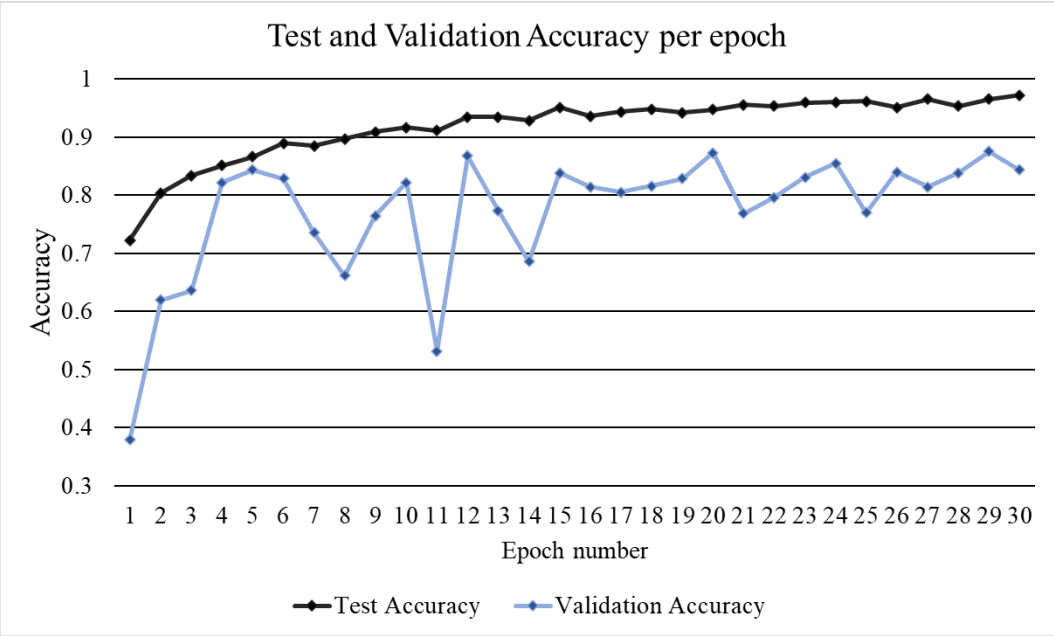

Figure 3.5: Test and Validation Accuracy per epoch using the fine-tuning technique. Source: Author

In the contemporary literature, an amalgamated approach that blends principles of both fine-tuning and transfer learning has been proposed. This hybrid method entails an initial phase where the model's final layer undergoes training in a transfer learning fashion for several epochs. Subsequent to this, a selective fine-tuning phase commences wherein only the concluding convolutional layers of the model are trained, effectively tailoring the higher-level feature representations to the specific dataset. In this study, this combined strategy was implemented maintaining consistent parameters with prior experiments. The training commenced with an exclusive 15-epoch training of the final layer, succeeded by another 15-epoch training that encompassed the last three convolutional blocks of the ResNet50 architecture. The empirical results of this experiment are depicted in Figure 3.6. In the conducted experiments, the optimal performance was achieved by the model trained on the 27th epoch, showcasing a validation accuracy of 91.18%. This result demonstrates a noteworthy enhancement in performance compared to either fine-tuning or transfer learning implemented independently over an equivalent number of training epochs.

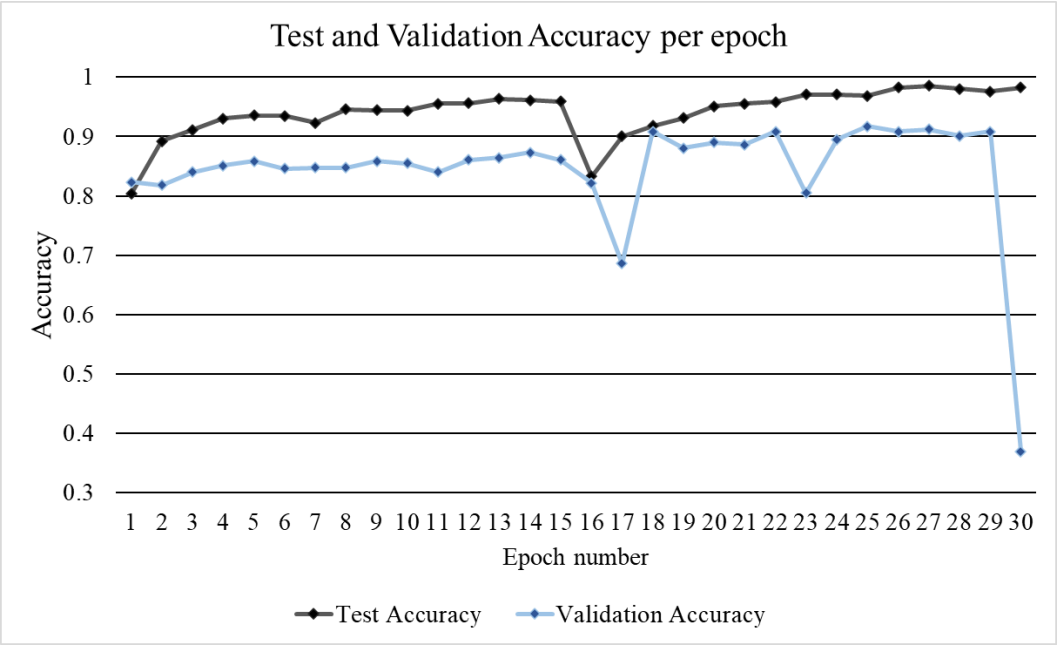

Figure 3.6: Test and Validation Accuracy per epoch using the hybrid technique. Source: Author

In Table 3.4, a comparative analysis of the peak validation accuracies across transfer learning, fine tuning, and the hybrid method is presented. Evidently, the hybrid approach exhibits superior performance, registering the highest prediction accuracy on the validation dataset relative to the other methodologies.

Table 3.4: Best validation accuracy results for each of the techniques.

Technique	Number of epochs	Test Accuracy	Validation Accuracy
Transfer learning	15	0.9469	0.8750
Fine tuning	30	0.9657	0.8750
Hybrid	30	0.9859	0.9118

In the context of detecting damages in wind turbine blades, the constraints imposed by the limited dataset necessitated adjustments to the CNN architecture to reduce the likelihood of overfitting. Initially, the model was structured to classify various categories and degrees of damage. Despite integrating several regularization methods discussed in the preceding chapter, such as weight decay, early stopping, data augmentation, and hyperparameter tuning, overfitting challenges persisted. To address this, the model's objective was pivoted towards a binary damage detection framework. This transition involved manually reclassifying the dataset, adapting the loss function to binary cross entropy loss and changing the activation function of the output layer to sigmoid, these procedures maximize the available data for learning in each step while simplifying the problem. The regularization techniques were subsequently reapplied to this revised model structure. In Figure 3.7, the ultimate architecture of the model is depicted.

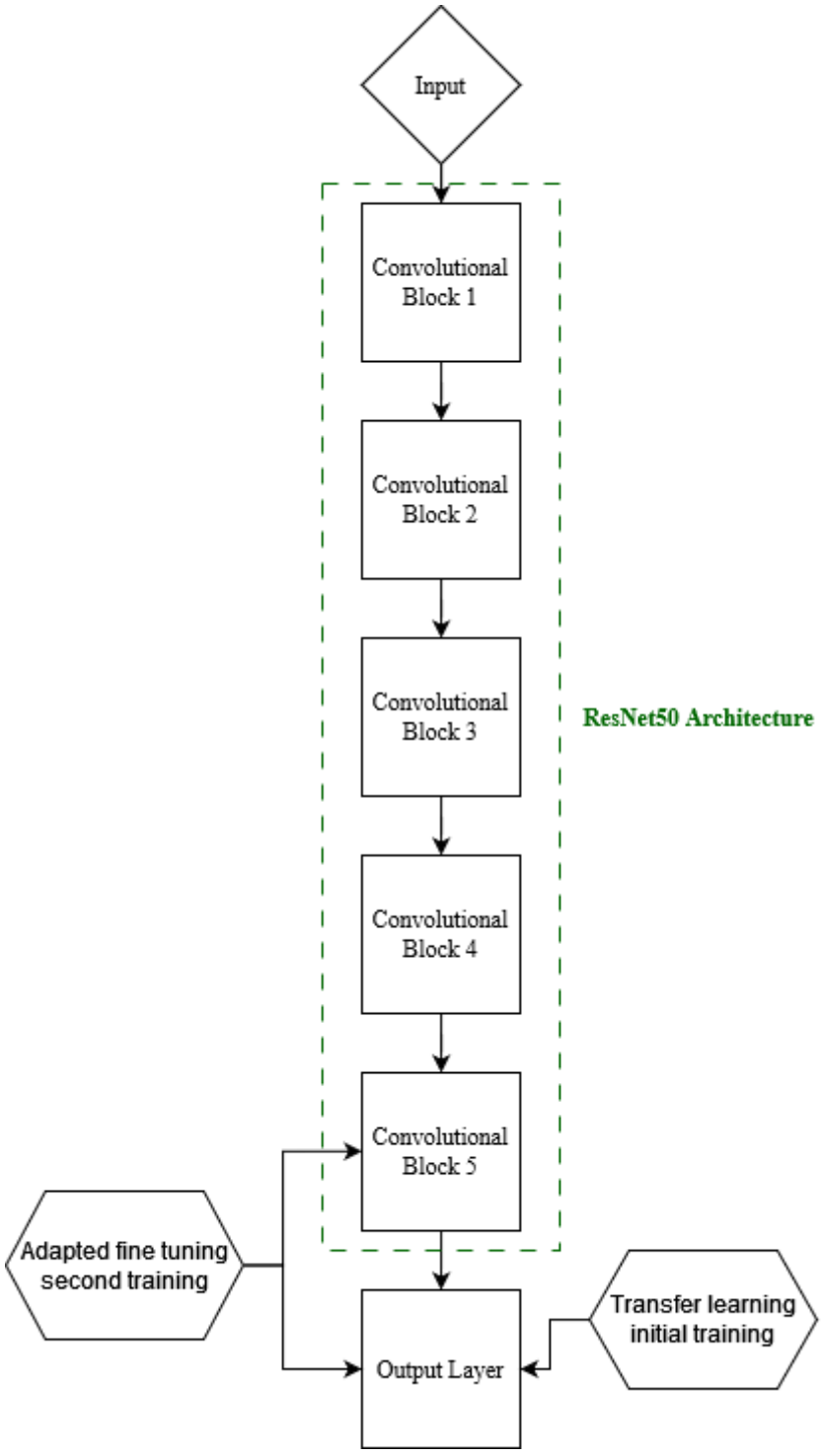

Figure 3.7: Final Architecture of the CNN used for defect detection. Source: Author

This is essentially a streamlined version of the ResNet50, wherein numerous convolutional layers have been amalgamated into distinct convolutional blocks, the structure of a single convolutional block is shown in Figure 3.8. Notably, during the second step of the hybrid training approach, the final convolutional block was subjected to training.

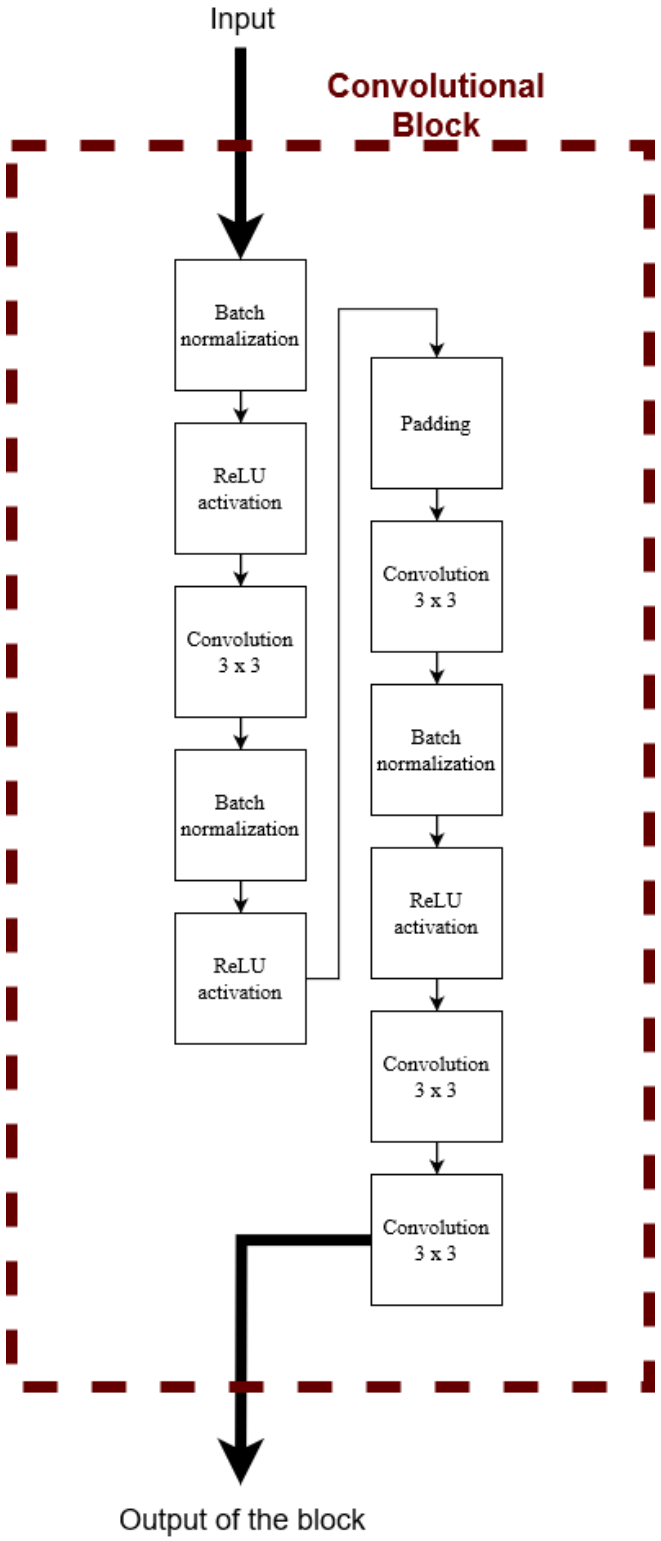

Figure 3.8: Structure of a single convolutional block in the ResNet50 architecture. Source: Author

THIS PAGE WAS INTENTIONALLY LEFT BLANK

4. RESULTS AND ANALYSIS

The objective of this chapter is to detail the outcomes derived from the training of the devised convolutional neural network, aimed at identifying damage in wind turbine blades. This section consists in the series of CNN architecture and hyperparameter iterations, shedding light on the alterations made to hyperparameters in each subsequent training exercise. Following the exposition of each model, it is provided an analytical discussion on its results, leading to the introduction of ensuing hyperparameter modifications.

4.1. CNN approach and hyperparameter selection

In the first proposed CNN architecture for identifying defects in wind turbine blade images, the primary objective was to craft a model proficient in detecting a variety of damage types and further categorizing them into specific categories. This challenge falls under the standard multi-class categorization paradigm. The suitable loss function for this task, categorical cross entropy, was chosen and the corresponding softmax activation function for the architecture's output layer was consequently defined.

A common technique to find the best set of hyperparameters for a given CNN architecture to maximize the accuracy of the predictions is called grid search. In grid search the training of the model is repeated with different specified hyperparameters, to apply this technique first, you specify a set of values for each hyperparameter you want to tune. Essentially, the model undergoes training for every possible permutation of the provided hyperparameters. To put this into perspective, the total number of unique training iterations required corresponds to the product of the number of values stipulated for each hyperparameter. After the completion of all training iterations, grid search yields the hyperparameter set that demonstrated the finest performance on the validation set. In the present study, the hyperparameters subjected to tuning via grid search included the optimizer's learning rate (η), momentum parameter (β_1), adaptive learning rate parameter (β_2), and the weight decay regularization parameter (λ). Table 4.1 delineates the potential values designated for each of these hyperparameters, resulting in the training of 36 distinct models for the comprehensive execution of the grid search process.

Table 4.1: Parameters for the grid search executed on the first CNN architecture.

Hyperparameter	**Value 1**	**Value 2**	**Value 3**
learning rate (η)	0.01	0.001	0.0001
momentum parameter (β_1)	0.9	0.99	-
adaptive learning rate parameter (β_2)	0.99	0.999	-
weight decay parameter (λ)	0	0.00001	0.0001

This approach yields a series of models to be trained, each tailored with distinct hyperparameters. Table 4.2 describes the various models generated by the grid search. Specifically, each unique combination of hyperparameter values is manifested as a distinct model.

Table 4.2: Models trained for the grid search on the first CNN architecture.

Model	η	β_1	β_2	λ	Loss function	Activation function	Epochs
Model 1	0.01	0.9	0.99	0	Categorical cross entropy	softmax	30
Model 2	0.01	0.9	0.99	0.00001	Categorical cross entropy	softmax	30
Model 3	0.01	0.9	0.99	0.0001	Categorical cross entropy	softmax	30
Model 4	0.01	0.9	0.999	0	Categorical cross entropy	softmax	30
Model 5	0.01	0.9	0.999	0.00001	Categorical cross entropy	softmax	30
Model 6	0.01	0.9	0.999	0.0001	Categorical cross entropy	softmax	30
Model 7	0.01	0.99	0.99	0	Categorical cross entropy	softmax	30
...	...	...	...	...	...	...	...
Model 36	0.0001	0.99	0.999	0.0001	Categorical cross entropy	softmax	30

In the second proposed CNN architecture for identifying defects in wind turbine blade images, the primary objective was to craft a model proficient in detecting the presence or absence of damage. This challenge falls under the standard binary categorization paradigm. The suitable loss function for this task, binary cross entropy, was chosen and the corresponding sigmoid activation function for the architecture's output layer was consequently defined. In a subsequent phase, another grid search was executed, this time with a primary focus on bolstering the model's regularization to combat overfitting. The hyperparameters designated for this grid search mirrored those selected for the initial CNN architecture. The specific values adopted for this grid search are shown in Table 4.3, resulting in the training of another 36 distinct models for the comprehensive execution of the grid search process. In this grid search iteration, a notable alteration was made: all learning rate parameters were decreased by an order of magnitude, and

concurrently, all weight decay parameters were augmented by an order of magnitude. This adjustment was strategized to maximize the chance of achieving at least one training epoch that is devoid of overfitting.

Table 4.3: Parameters for the grid search executed on the second CNN architecture.

Hyperparameter	**Value 1**	**Value 2**	**Value 3**
learning rate (η)	0.001	0.0001	0.00001
momentum parameter (β_1)	0.9	0.99	-
adaptive learning rate parameter (β_2)	0.99	0.999	-
weight decay parameter (λ)	0.00001	0.0001	0.001

The models are numbered following the previous grid search in order to maintain clarity on which CNN architecture is being analysed. The multi-class CNN architecture designed for damage classification is represented by models 1-36 while the binary CNN architecture designed for damage detection is represented by models 37-72. Table 4.4 describes the various models generated by the grid search on the second architecture.

Table 4.4: Models trained for the grid search on the second CNN architecture.

Model	η	β_1	β_2	λ	Loss function	Activation function	Epochs
Model 37	0.001	0.9	0.99	0.00001	Binary cross entropy	sigmoid	30
Model 38	0.001	0.9	0.99	0.0001	Binary cross entropy	sigmoid	30
Model 39	0.001	0.9	0.99	0.001	Binary cross entropy	sigmoid	30
Model 40	0.001	0.9	0.999	0.00001	Binary cross entropy	sigmoid	30
Model 41	0.001	0.9	0.999	0.0001	Binary cross entropy	sigmoid	30
Model 42	0.001	0.9	0.999	0.001	Binary cross entropy	sigmoid	30
Model 43	0.001	0.99	0.99	0.00001	Binary cross entropy	sigmoid	30
…	…	…	…	…	…	…	…
Model 72	0.00001	0.99	0.999	0.001	Binary cross entropy	sigmoid	30

4.2. Results

The outcomes of the first proposed CNN architecture are evident when examining the training results of Model 1, as presented in Table 4.5. Notably, in the inaugural training epoch, the model attains a 100 percent accuracy on both the training and validation data. Such a phenomenon is typical in scenarios with limited datasets. Rather than assimilating the intrinsic features and overarching patterns of the data, the model tends to overfit, essentially memorizing specific data points from the dataset. Models 2-36 exhibit identical shortcomings to that of Model 1. This recurrent issue signifies a pervasive challenge in the training phase. Given these consistent anomalies, the trained CNNs from this series are unlikely to offer dependable predictions on new, unobserved data.

Table 4.5: Training of Model 1 where omitted epochs were identical.

Epoch	Time	Test Accuracy	Validation Accuracy
1	30 seconds	1	1
2	17 seconds	1	1
3	17 seconds	1	1
...	...	...	...
28	17 seconds	1	1
29	17 seconds	1	1
30	17 seconds	1	1

The second CNN was proposed to maximize the use of training data while minimizing the complexity of the problem in order to mitigate the overfitting. The hyperparameters selected for use in the grid search were also selected to maximize the regularization of the model while still allowing for training, this was done in order to try to train a useful model from the dataset available. However as can be seen in Table 6 which displays the results of the training of Model 72, even the model with the most severe regularization and the lowest learning rates displayed instantaneous overfitting to the dataset. The issues encountered are primarily attributed to the limited size of the dataset utilized in this study. For context, the smallest dataset reported in existing literature for training a CNN specifically for defect detection in WTBs is approximately 17 times larger than the dataset employed in this research. The sheer disparity in data volume underscores the challenges faced and underscores the critical role of a sufficiently large dataset in successfully training a neural network.

Table 4.6: Training of Model 72 where omitted epochs were identical.

Epoch	Time	Test Accuracy	Validation Accuracy
1	29 seconds	1	1
2	15 seconds	1	1
3	15 seconds	1	1
...	...	...	...
28	15 seconds	1	1
29	15 seconds	1	1
30	15 seconds	1	1

Given that the proposed CNN architecture demonstrated efficacy in multi-class classification tasks with a more expansive database, as discussed in Section 3, and considering that all implemented strategies to counteract overfitting proved insufficient for the available data, the primary takeaway from this section underscores the imperative need for a more comprehensive dataset to effectively train a CNN capable of producing reliable predictions. The second proposed CNN architecture is inherently designed to be more adaptable to smaller datasets compared to the first due to the reduced complexity of its output expectations. However, its utility for industrial applications might be perceived as limited. The architecture only signals the existence of damage, necessitating further manual inspection by experts to classify the nature and severity of the detected damage. In contrast, given an ample dataset, the first proposed architecture has the potential to not only identify but also categorize the damage in terms of its type and severity. This capability offers a more automated and comprehensive solution for the damage inspection process, making it a more desirable option in the long run.

THIS PAGE WAS INTENTIONALLY LEFT BLANK

5. CONCLUSIONS AND FURTHER DEVELOPMENTS

5.1. Main Conclusions

A comprehensive review of the literature concerning defects in wind turbine blades (WTB) was undertaken, focusing on those imperfections that notably impact the lifespan of the turbines. The principal inspection methodologies deployed for WTB include visual inspection, which primarily targets surface damages like cracks or leading-edge erosion. Digital shearography is employed to detect subsurface anomalies, notably delaminations, while thermography is used for identifying subsurface porosities, cracks, and for real-time monitoring of defect progression. A shared challenge across these inspection modalities is the generation of substantial volumes of data, which not only escalates the associated human labour costs but also introduces potential for errors in analysis.

Convolutional neural networks (CNN) have emerged as the state-of-the-art methodology for classifying damages utilizing vast amounts of data. An exhaustive review of prevalent architectures in the literature was undertaken, with ResNet50 emerging as a standout, offering an optimal blend of simplicity and efficacy for handling both extensive and limited datasets. While transfer learning and fine-tuning are frequently cited in literature, the hybrid approach, which merges elements of both, remains less explored. Interestingly, when testing the architecture on a non-related dataset this hybrid technique showcases superior training outcomes within the same training epoch count.

Data augmentation was implemented on the received wind turbine blade (WTB) inspection data; however, the dataset size remained inadequate to train a multi-class CNN architecture. To mitigate this, a revised CNN architecture was introduced, focusing on binary categorization. This adjustment aimed to simplify the problem and leverage the available training data optimally, sidestepping class imbalance issues. However, even this binary-focused CNN architecture encountered training difficulties. This confirms the insufficiency of the dataset in hand, aligning with literature findings. Notably, the smallest dataset known to have successfully trained a damage detection CNN is significantly more extensive than the current dataset by an order of magnitude.

5.2. Further Development

The realm of defect inspection within the wind energy sector is witnessing rapid advancements, with machine learning emerging as the premier approach for deciphering the vast amounts of data generated by modern inspection techniques. In light of the insights from this research, the following avenues are proposed for future exploration:

- Procure and analyse a more extensive dataset to ascertain the minimal data prerequisites for training both binary and multi-class CNNs effectively;
- Harness a more comprehensive dataset specific to wind turbine blade inspections for the successful training of a CNN tailored for multi-class damage detection;
- Utilize data from shearography to train a CNN to detect sub-surface defects.

REFERENCES

[1] W. Zhong, H. An, L. Shen, W. Fang, X. Gao, and D. Dong, 'The roles of countries in the international fossil fuel trade: An emergy and network analysis', *Energy Policy*, vol. 100, pp. 365–376, Jan. 2017, doi: 10.1016/j.enpol.2016.07.025.

[2] M. Cheng and Y. Zhu, 'The state of the art of wind energy conversion systems and technologies: A review', *Energy Conversion and Management*, vol. 88. Elsevier Ltd, pp. 332–347, 2014. doi: 10.1016/j.enconman.2014.08.037.

[3] V. M. Karbhari *et al.*, 'Contributor contact details', in *Non-Destructive Evaluation (NDE) of Polymer Matrix Composites*, V. M. Karbhari, Ed., Woodhead Publishing, 2013, pp. xv–xx. doi: https://doi.org/10.1016/B978-0-85709-344-8.50028-2.

[4] L. Zou and H. Cheng, 'Research on Wind Turbine Blade Surface Damage Identification Based on Improved Convolution Neural Network', *Applied Sciences (Switzerland)*, vol. 12, no. 18, Sep. 2022, doi: 10.3390/app12189338.

[5] Lee Joyce and Zhao Feng, 'GWEC-Global-Wind-Report-2021', Brussels, Belgium, Mar. 2021.

[6] J. Thé and H. Yu, 'A critical review on the simulations of wind turbine aerodynamics focusing on hybrid RANS-LES methods', *Energy*, vol. 138. Elsevier Ltd, pp. 257–289, 2017. doi: 10.1016/j.energy.2017.07.028.

[7] K. Shao, Y. He, Z. Xing, and B. Du, 'Detecting wind turbine anomalies using nonlinear dynamic parameters-assisted machine learning with normal samples', *Reliab Eng Syst Saf*, vol. 233, p. 109092, May 2023, doi: 10.1016/J.RESS.2023.109092.

[8] World Meteorological Organization, *Meteorological Aspects of the Utilization of Wind as an Energy Source*. in Note technique / Organisation meteorologique mondiale. Secretariat of the World Meteorological Organization, 1981. [Online]. Available: https://books.google.com.br/books?id=S5kRAQAAIAAJ

[9] G. Gaudiosi, 'Offshore wind energy prospects', *Renew Energy*, vol. 16, no. 1, pp. 828–834, 1999, doi: DOI: 10.1016/S0960-1481(98)002,.

[10] M. F. Akorede, 'Design and performance analysis of off-grid hybrid renewable energy systems', *Hybrid Technologies for Power Generation*, pp. 35–68, Jan. 2022, doi: 10.1016/B978-0-12-823793-9.00001-2.

[11] E. Z. Uriguen, 'Design and validation of a methodology for wind energy structures health monitoring', 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:190074833

[12] N. Stavridou, E. Efthymiou, S. Gerasimidis, and C. Baniotopoulos, *improvement of steel wind turbine tower structural response with implementation of steel stiffening rings*. 2014.

[13] E. Sun, 'Spinning electricity out of thin, moving air - Science in the News', Dec. 15, 2012. https://sitn.hms.harvard.edu/flash/2012/wind/ (accessed Jun. 28, 2023).

[14] L. Mishnaevsky, K. Branner, H. N. Petersen, J. Beauson, M. McGugan, and B. F. Sørensen, 'Materials for wind turbine blades: An overview', *Materials*, vol. 10, no. 11. MDPI AG, Nov. 01, 2017. doi: 10.3390/ma10111285.

[15] Ever J. Barbero, *Introduction to Composite Materials Design*, vol. 1. CRC Press, 2017.

[16] A. Brent. Strong, *Fundamentals of composites manufacturing : materials, methods and applications*. Society of Manufacturing Engineers, 2008.

[17] L. Zou, Y. Wang, J. Bi, and Y. Sun, 'Damage Detection in Wind Turbine Blades Based on an Improved Broad Learning System Model', *Applied Sciences*, vol. 12, no. 10, 2022, doi: 10.3390/app12105164.

[18] J.-S. Chou, C.-K. Chiu, I.-K. Huang, and K.-N. Chi, 'Failure analysis of wind turbine blade under critical wind loads', *Eng Fail Anal*, vol. 27, pp. 99–118, Jan. 2013, doi: 10.1016/j.engfailanal.2012.08.002.

[19] L. Mishnaevsky Jr. and K. Thomsen, 'Costs of repair of wind turbine blades: Influence of technology aspects', *Wind Energy*, vol. 23, no. 12, pp. 2247–2255, 2020, doi: https://doi.org/10.1002/we.2552.

[20] S. J. Park and M. K. Seo, 'Types of Composites', *Interface Science and Technology*, vol. 18, pp. 501–629, Jan. 2011, doi: 10.1016/B978-0-12-375049-5.00007-4.

[21] K. K. Chawla, 'Composite Materials Science and Engineering'. Springer, 1987.

[22] M. Gopi Krishna, K. Praveen Kumar, M. Naga Swapna, J. Babu Rao, and N. R. M. R. Bhargava, 'Metal-metal Composites-An Innovative Way For Multiple Strengthening', *Mater Today Proc*, vol. 4, no. 8, pp. 8085–8095, Jan. 2017, doi: 10.1016/J.MATPR.2017.07.148.

[23] D. P. Gonçalves, F. C. L. Melo, A. Klein, and H. Al-Qureshi, 'Analysis and investigation of ballistic impact on ceramic/metal composite armour', *Int J Mach Tools Manuf*, vol. 44, pp. 307–316, Feb. 2004, doi: 10.1016/j.ijmachtools.2003.09.005.

[24] F. L. Matthews and R. D. Rawlings, *Composite Materials: Engineering and Science*. in Composite Materials: Engineering and Science. CRC Press, 1999. [Online]. Available: https://books.google.pt/books?id=0p4I5VRJmrsC

[25] S. G. Advani and K.-T. Hsiao, '1 - Introduction to composites and manufacturing processes', in *Manufacturing Techniques for Polymer Matrix Composites (PMCs)*, S. G. Advani and K.-T. Hsiao, Eds., in Woodhead Publishing Series in Composites Science and Engineering. Woodhead Publishing, 2012, pp. 1–12. doi: https://doi.org/10.1533/9780857096258.1.1.

[26] B. F. Sørensen *et al.*, 'Improved design of large wind turbine blade of fibre composites based on studies of scale effects (Phase 1)', 2004, [Online]. Available: www.risoe.dk

[27] J. Bai, Ed., 'Woodhead Publishing Series in Civil and Structural Engineering', in *Advanced Fibre-Reinforced Polymer (FRP) Composites for Structural Applications*, Woodhead Publishing, 2013, pp. xix–xxii. doi: https://doi.org/10.1016/B978-0-85709-418-6.50026-0.

[28] R. B. Heslehurst, 'Defects and Damage in Composite Materials and Structures', *Defects and Damage in Composite Materials and Structures*, Apr. 2014, doi: 10.1201/B16765.

[29] A. SHIHAVUDDIN and X. Chen, 'DTU - Drone inspection images of wind turbine', vol. 2, 2018, doi: 10.17632/HD96PRN3NC.2.

[30] Z. Li, M. O. Tokhi, Z. Zhao, and H. Zheng, 'A Compact Laser Shearography System for On-Site Robotic Inspection of Wind Turbine Blades', *Journal of Artificial Intelligence and Technology*, vol. 1, no. 3, pp. 166–173, Jul. 2021, doi: 10.37965/JAIT.2021.0008.

[31] S. Sfarra *et al.*, 'A comparative investigation for the nondestructive testing of honeycomb structures by holographic interferometry and infrared thermography', *J Phys Conf Ser*, vol. 214, no. 1, p. 012071, Mar. 2010, doi: 10.1088/1742-6596/214/1/012071.

[32] X. Chen, A. S. M. Shihavuddin, S. H. Madsen, K. Thomsen, S. Rasmussen, and K. Branner, 'AQUADA: Automated quantification of damages in composite wind turbine blades for LCOE reduction', *Wind Energy*, vol. 24, no. 6, pp. 535–548, 2021, doi: https://doi.org/10.1002/we.2587.

[33] A. Reddy, V. Indragandhi, L. Ravi, and V. Subramaniyaswamy, 'Detection of Cracks and damage in wind turbine blades using artificial intelligence-based image analytics', *Measurement (Lond)*, vol. 147, Dec. 2019, doi: 10.1016/j.measurement.2019.07.051.

[34] A. P. Marugán, F. P. G. Márquez, J. M. P. Perez, and D. Ruiz-Hernández, 'A survey of artificial neural network in wind energy systems', *Applied Energy*, vol. 228. Elsevier Ltd, pp. 1822–1836, Oct. 15, 2018. doi: 10.1016/j.apenergy.2018.07.084.

[35] Fitchett Brandon, 'A WHITE PAPER ON WIND TURBINE BLADE DEFECT AND DAMAGE CATEGORIZATION', Palo Alto, California, 2020. Accessed: Sep. 03, 2023. [Online]. Available: https://www.epri.com/research/products/000000003002019669

[36] A. Géron, 'Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow SECOND EDITION Concepts, Tools, and Techniques to Build Intelligent Systems'.

[37] J. Patterson and A. Gibson, 'Deep Learning'.

[38] S. Raschka and V. Mirjalili, *Python machine learning : machine learning and deep learning with python, scikit-learn, and tensorflow 2*.

[39] C. C. Aggarwal, 'Neural Networks and Deep Learning', *Neural Networks and Deep Learning*, 2018, doi: 10.1007/978-3-319-94463-0.

[40] C. Quandel, V. Schmid, and L. München, 'Deep Convolution Neural Networks for the Analysis of a few Medical Images Eidesstattliche Erklärung'.

[41] Y. LeCun *et al.*, 'Backpropagation Applied to Handwritten Zip Code Recognition', *Neural Comput*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/NECO.1989.1.4.541.

[42] A. C. Müller and S. Guido, 'Introduction to Machine Learning with Python A GUIDE FOR DATA SCIENTISTS Introduction to Machine Learning with Python'.

[43] A. Iyer, L. Nguyen, and S. Khushu, 'Learning to identify cracks on wind turbine blade surfaces using drone-based inspection images'.

[44] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, 'A Survey of Deep Neural Network Architectures and Their Applications'.

[45] L. N. Smith, 'A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay', *CoRR*, vol. abs/1803.09820, 2018, [Online]. Available: http://arxiv.org/abs/1803.09820

[46] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Research, G. (n.d.). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved September 4, 2023, from www.tensorflow.org.