Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Jan Neidhöfer

Computer Vision for Defect Detection in Wood Manufacturing

Master's thesis in Industrial Engineering and Management
Supervisor: Geir Ringen
Co-supervisor: Stine Brenden Bjurlemyr
August 2022

NTNU
Norwegian University of
Science and Technology

Jan Neidhöfer

Computer Vision for Defect Detection in Wood Manufacturing

Master's thesis in Industrial Engineering and Management
Supervisor: Geir Ringen
Co-supervisor: Stine Brenden Bjurlemyr
August 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Wood is a sought-after resource used in various application areas, representing sustainability and natural aesthetics. However, the wood industry and its secondary wood products face challenges of low levels of automation and quality issues. Due to high material costs, improvements in utilization are necessary to remain competitive in a volatile environment. This goal cannot be achieved through traditional manual quality controls. Computer vision approaches for defect detection in the wood industry hold great potential. Automated non-destructive technologies (NDT) can significantly enhance operational efficiency and lay the foundation for zero-defect manufacturing (ZDM) principles. In the context of this work, the effective application of image classification and object detection computer vision technologies for wood defect detection is examined. Furthermore, this thesis explores whether offline data augmentation and transfer learning are effective methods for improving performance with limited data quantities. To further evaluate these methods, new data is collected using a self-installed camera system in a simulated production environment. The results demonstrate that modern YOLOv7 and YOLOv8 one-stage object detectors outperform classic image classification algorithms in terms of overall performance and usability. Through the application of offline data augmentation and transfer learning, performance can be partly enhanced when working with limited data sets.

Contents

Chapter 1

Introduction

1.1 Motivation

Wood, a fundamental natural resource and has been indispensable to human civilization for centuries. The material is widely used in construction and in everyday life, e.g. in the home as furniture or decoration, as musical instruments or as luxury material in cars. Due to its exclusive mechanical properties and it's aesthetic appeal, the material is a demanded resource in modern times. As a result, the industry has undergone a resurgence in importance. The increasing climate awareness of consumers and investors as well as political pressure and social calls for more energy-efficient production, renewable materials, and circular economy also contribute to this. Lower emissions from the processing of wood products and the material's natural function of storing atmospheric carbon dioxide over the long term are boosting the market potential of wood and derived production.

On the other hand, the changes in the market and climate present new challenges to the wood processing industry. The cost of raw material is rising, as are consumer demands for product quality and customizability according to their own wishes. As a result, accurate and consistent defect detection is a pivotal challenge in maintaining the quality of wood products. An industry that today is still often characterized by manual work, inefficient processes and low utilization rates must fundamentally adapt its strategy in order not to collapse in the face of local and global competition. Human-centric defect detection as an important part of quality assurance cannot stand up to today's industrial demands because of its serious disadvantages.

Herein lies the urgency for the wood products industry to adapt its production processes and quality control to new standards by means of transformative, digital concepts of the Industry 4.0 paradigm. Technology-enabled production strategies such as zero-defect manufactur-

ing are important drivers for cost reductions and resource-efficient production and underscore the need for innovative approaches that can seamlessly integrate into the wood manufacturing pipeline.

The integration of computer vision for defect detection within wood manufacturing holds significant potential. By improving defect detection accuracy and automating quality control, the efficiency within the industry can be elevated to match the modern manufacturing landscape. Transfer learning offers new opportunities for small and medium-sized enterprises with a small data base and can mitigate the problem of data protection concerns.

1.2 Research Questions

This work aims to clarify whether computer vision methods are a suitable method for detecting surface defects in the wood products industry under the paradigm of zero-defect manufacturing (ZDM). To determine the potential of the methods, algorithms for multi-label image classification and state-of-the-art object detection frameworks are evaluated and benchmarked regarding their performance for detecting and localizing defects on wooden surfaces. A publicly available data set of wooden surfaces is utilized for this analysis. In addition, it is tested whether offline data augmentation and transfer learning can provide advantages for object detection in the domain of wood defect detection where large amounts of data are typically not available. By means of a use case, the transferability to a real production environment in a manufacturing plant for the production of wooden windows will be simulated. For this, a prototype camera setup is installed in an Industry 4.0 learning factory environment. This setup is utilized to acquire image data of wooden surfaces to simulate application in a real production environment. The acquired data is then evaluated with the models and methods. The results of the tests will be used to determine whether it is suitable to install and apply the technology in production operations.

The following research questions are addressed within this work:

- **Research question 1:** How can computer vision techniques be effectively applied to enhance the defect detection process in wood manufacturing processes?
- **Research question 1a:** How well suited are ResNet and EfficientNet models for defect detection on wooden surfaces based on their performance?
- **Research question 1b:** How well suited are YOLOv7 and YOLOv8 object detection models for defect detection and localization on wooden surfaces based on their performance?

- **Research question 2:** How does offline data augmentation impact the training and performance of object detection models?
- **Research question 3:** How does transfer learning impact the performance of object detection models in the context of defect detection on wooden surfaces?
- **Research question 3a:** How effective is transfer learning based on pre-training on generic image databases?
- **Research question 3b:** How effective is transfer learning based on pre-training on a different data set of wood surfaces?

1.3 Outline

This thesis is structured as follows: Chapter 1 introduces the thesis with a short introductory motivation and presents the research questions of this work. Chapter 2 deals with important fundamentals related to the wood industry as well as fundamental concepts. The wood defects under investigation are also explained, and the necessity of automated defect detection is outlined. Subsequently, Chapter 3 explains the relevant technical and theoretical foundations of deep learning and computer vision, which form the basis of this work. Chapter 4 introduces related literature. Chapter 5 describes the used data and the methodological approach to address the research questions. In Chapter 6, information about the use case is provided, and the associated data acquisition process is explained before analyzing the collected data. Chapter 7 presents the results of the conducted experiments, followed by a discussion in Chapter 8.1. Finally, the thesis concludes with chapter 9.

Chapter 2

Background and Foundations

2.1 Background

2.1.1 Wood as a Material

Forests offer a variety of functions, including but not limited to their contribution to the global carbon cycle, provision of usable material and energy, and preservation of the biological diversity (Butarbutar et al. 2016). About 50 % of the forests in our world are involved in the generation of forest products, which include both wood and non-wood items (Ramage et al. 2017). Around half of the wood that is harvested globally is employed for industrial purposes and undergoes processing to produce wood products. The other half is utilized as a source of energy (Kromoser et al. 2022). Due to its widespread availability and renewability, wood is an essential resource that has been used for hundreds of years for building refuges and as an asset for various industrial processes (Ross 2015). Despite the replacement of many of wood's previous applications by innovative alternative materials, wood is is currently experiencing a resurgence in its significance (Arriaga et al. 2023).

As a natural resource, wood and it's derivative products are widely used in our lives, e.g. for furniture, construction, or decoration. Hence, the material plays a significant role in today's manufacturing industry (Li et al. 2021, Molinaro & Orzes 2022, Gao et al. 2022, Ding et al. 2020). In addition to its visual appeal, the material has positive properties in terms of resistance and elasticity, and can therefore be used in a variety of applications (Yu et al. 2019, Ding et al. 2020).

Important parameters for measuring the mechanical properties of wood are the modulus of rupture (MOR) and the modulus of elasticity (MOE). The former describes the indicates the bending strength along the grain direction, while the latter describes the deformation behav-

ior under load (Muñoz & Gete 2012). The material has anisotropic properties, i.e. its physical and mechanical properties vary significantly along different axes. This is mainly due to the arrangement of fibers in longitudinal direction. The properties in the direction that follows the length of the wood fibers differ significantly to those perpendicular to the length of the fibers (Ross 2015, Zielińska & Rucka 2021). MOR and MOE experience a notable increase when transitioning from the pith towards the bark of the wood (Habite et al. 2022, 2021). Wood comprises cellulose, lignin, and hemicellulose as its key components. Its distinctive diversity in properties sets it apart from other materials (Zielińska & Rucka 2021). Due to its qualities, wood is used for a variety of structural and nonstructural applications in the construction industry (Ahn & Park 2020). Examples for the non structural use of wood are window frames, doors, and decoration. Window frames made from wood are among the most popular options in Europe and North-America (Ahn & Park 2020).

2.1.2 Environmental Impacts and Market Growth of Wood Industry

Climate change, the raising awareness for environmentally friendly products, and the overall increased availability for wooden products indicate growth of the wood industry (Heräjärvi et al. 2019), and the market for high-quality wood products (Lin & Sanjaya 2021). It is anticipated that there will be greater variety in the forest product markets in the future (Leskinen et al. 2018). It has been shown, that using wood-based products helps alleviate climate change (Smyth et al. 2016). A part of this mitigation results from the substitution of materials and energy (Kayo et al. 2015). Several studies have demonstrated that greenhouse gas emissions caused by wood-based products are lower than those from fossil-based alternatives (Seppälä et al. 2019). The energy input during the production process of wood based products is typically lower than for functionally comparable materials (Butarbutar et al. 2016). The advantages of substitution mainly arise from the decreased emissions during both the production phase and the end-of-life phase (Leskinen et al. 2018). In addition, products made from harvested wood have the ability to store carbon that has been absorbed in the form of carbon dioxide from the atmosphere on a long-term basis. Recycling these products at the end of their life cycle maintains extends the duration of carbon storage (Geng et al. 2017). According to (Broda 2020), wood is holding the largest terrestrial reservoir of stored carbon. Consequently, in the building industry, which is characterized by significant greenhouse gas emissions, investors have redirected their focus towards using wood as a substitute due to its function as carbon storage (Arriaga et al. 2023). Furthermore, for the reduction of waste in the manufacturing sector the European Union's circular economy strategy is a driver as it incorporates a stricter waste legislation (Heräjärvi et al. 2019). The political pressure for environmentally friendly production in combination with technological process forces producing companies to focus on improvements of their manufacturing processes (Eric-

sson et al. 2021).

2.1.3 Industry Needs

Economies are becoming more unpredictable and there is a trend towards tailoring products to individual needs. Therefore, production processes need to be flexible to meet these demands, while maintaining consistent quality and efficient control (Eger et al. 2018). With regard to the wood and wood-processing industry, material costs for raw material has been increasing due to unregulated deforestation and the resulting scarcity of resources have led to high material costs making up to 70% of the costs for secondary wooden products (Chun et al. 2023, Gao, Qi, Mu & Chen 2021, Hashim et al. 2015). In addition, quantitative requirements are often not met due to environmental influences or slow growth rates of trees (Ding et al. 2020). To overcome these issues, companies use materials of lower quality (Kline et al. 2003). The requirements for loss-free processing of the products have risen sharply in the process. For example, the material utilization rate in northern European production countries is up to 90 %, while in Asia a maximum of 60 % is achieved in some cases (Yang et al. 2020). Moreover, the industry is centered on manual activities and exhibits limited levels of automation as well as a comparatively low productivity (Landscheidt & Kans 2016). In light of these difficulties, the wood products industry must actively engage in the pursuit of innovative technologies in order to thrive in a landscape that is becoming more and more competitive (Bond et al. 1998). In regions with high production costs, enhancing the level of automation within manufacturing processes by implementing appropriate systems is a commonly adopted strategy (Landscheidt & Kans 2016).

2.2 Foundations

2.2.1 Industry 4.0

Industry 4.0, a phrase introduced by Kagermann et al. (2011), describes the idea of the continuous digital transformation and swift technological progress happening in various industries and societies. When applied to manufacturing, Industry 4.0 encompasses the concept of smart manufacturing systems, the interlinking of data, and the utilization of cyber-physical systems. This means a shift from the previously centralized approach towards a product-centered control. The product itself determines the next steps of processing based on relevant parameters. Sensors record these parameters and enable monitoring as well as intervention in the process in the event of malfunctions. The idea of Industry 4.0 is alternatively referred to as the fourth industrial revolution (Kagermann et al. 2011). Kagermann et al. (2013) claim that Industry 4.0

drives the emergence of innovative value creation and novel business models, creating opportunities for startups and small businesses. Furthermore, Industry 4.0 confronts urgent global challenges, including energy efficiency, resource efficiency, and demographic shifts. In the course of digital transformation, the scope is no longer limited to production itself, but encompasses the entire business organization (Vaidya et al. 2018, Issa et al. 2018, Ghobakhloo 2020). It drives ongoing improvements in productivity and resource efficiency throughout the entire value chain (Kagermann et al. 2013, Ghobakhloo 2020). By considering demographic changes and social factors, Industry 4.0 enables the organization of work in a manner that adapts to these transformations. For instance, intelligent assistance systems have the potential to alleviate workers from monotonous routine tasks, enabling them to redirect their efforts towards a wider range of activities that add value. This not helps aging employees but also addresses the impending shortage of skilled workers (Kagermann et al. 2013).

Although there is no universally accepted definition of Industry 4.0 within the industrial sector, the main building blocks of the concept can be named. These enablers include cyber-physical system (CPS) concepts and technologies such as cloud computing, Internet of Things (IoT), Internet of Services (IoS), additive manufacturing, big data, augmented reality, smart factories, cyber security, and autonomous robots (Rüßmann et al. 2015, Bumgardner & Buehlmann 2022, Hofmann & Rüsch 2017, Kagermann et al. 2013, Chen et al. 2018). However, it should be noted that the concepts are complementary and mutually enabling instead of operating independently (Ghobakhloo 2020). For example, the smart factory concept that is enabled by an interplay of IoT, IoS, and CPS (Hofmann & Rüsch 2017, Chen et al. 2018).

Kagermann & Wahlster (2022) identify artificial intelligence (AI), edge computing, 5G, collaborative robotics, autonomous intralogistics systems, and secure data infrastructures as the primary components driving the ongoing advancement of Industry 4.0 in the upcoming years. Some concepts are already increasingly implemented, such as industrial AI and edge computing for predictive maintenance systems. Newer concepts, such as AI-supported quality control, are also expected to play a major role in the future. The availability of camera systems, powerful GPUs and large amounts of digitally available data support the development. AI-assisted zero defect manufacturing (ZDM) is therefore cited as the industry's new paradigm. This enables the carving out of a competitive advantage over production locations with low labor costs and less technical expertise and goes hand in hand with the increased quality demands of customers (Kagermann & Wahlster 2022).

SMEs could benefit from the introduction of Industry 4.0 concepts and technologies and increase their competitiveness (Pech & Vrchota 2020). In many economies, SME make up the largest part of the business landscape and are often seen as the driving force of the economy (Stentoft et al. 2020, Mittal et al. 2018). However, the entry barrier for the implementation of

Industry 4.0 concepts is higher among SMEs. SMEs typically have fewer resources compared to large enterprises, lack requirement and have less capabilities to influence negotiations with suppliers (Bumgardner & Buehlmann 2022, Stentoft et al. 2020, Mittal et al. 2018, Müller 2019, Pech & Vrchota 2020).

Many of the secondary wood product firms fall into the category of SMEs. A study conducted by Bumgardner & Buehlmann (2022) shows that the majority of secondary wood product firms have limited knowledge about the concept of Industry 4.0. A different picture emerged when determining the automation or digitalization curve. On average, the companies surveyed recorded a medium level of automation. Around one third of the companies surveyed also stated that they had increasingly used digital technologies to optimize raw material processing over the past three years. A lack of financial resources and a shortage of skilled workers were cited as the biggest obstacles to the introduction of digital technologies (Bumgardner & Buehlmann 2022). A different picture regarding the wood products industry emerges according to the study published in 2016 by Landscheidt & Kans (2016). The Swedish wood and wood product industry, which makes up an important part of the Swedish economy and provides significant export values, lacks the necessary prerequisite for the implementation of automation and Industry 4.0 practices. However, the authors show through their research that this problem is not exclusive to Sweden, but that the industry in general is lagging behind in terms of its technical and organizational capabilities (Landscheidt & Kans 2016).

As an enhanced version of Industry 4.0, Industry 5.0 is emerging as a new vision with more focus on the employees, and future generations wellbeing (Demir & Cicibaş 2019). European Commission et al. (2021) define Industry 5.0 with three key factors: human-centric, sustainable, and resilient. However, the concept is not yet widely adapted in industry and many companies are still in the Industry 4.0 transformation process. Therefore, this work does not discuss the topic further, and the interested reader is referred to the referenced literature.

2.2.2 Zero Defect Manufacturing

In traditional manufacturing, the final product is checked for its condition according to predefined criteria in order to meet the quality requirements of customers. In case the product does not meet these requirements, time-consuming and resource-intensive rework becomes due, or the product is discarded completely (Eger et al. 2019). This is does not align with today's production demands and therefore requires new strategies (Raabe et al. 2018, Eger et al. 2019).

Zero-Defect Manufacturing (ZDM) refers to the manufacturing strategy with the goal of eliminating all defects from the manufacturing process (Psarommatis et al. 2019, Powell et al. 2022). The philosophy shifts the focus on the elimination of defects during the production process

rather than relying on post-production quality control measures (Gauder et al. 2023). The ultimate goal is to achieve the highest possible product quality while minimizing waste, rework, and customer complaints (Powell et al. 2022). In addition, cost savings and safety improvements can be realized (Psarommatis et al. 2019).

The technology-driven philosophy has gained significant prominence under the advent of Industry 4.0 and CPSs through the availability of sensors, powerful hardware, big data, and other associated technologies (Raabe et al. 2018, Psarommatis et al. 2019). These technologies enable companies to use proactive systems for detecting and preventing defects are employed in addition to six sigma and lean quality assurance and monitoring approaches (Powell et al. 2022, Gauder et al. 2023, Alexopoulos et al. 2023).

According to Powell et al. (2022), the ZDM philosophy goes hand in hand with the principles of sustainability and circular economy, which are becoming increasingly crucial for companies due to elevated requirements and customer awareness. The authors suggest that in addition to preventive measures for defect prevention, the focus should also be on the reuse of defective products and the implementation of non-destructive evaluation methods (Powell et al. 2022).

2.3 Wood Defects and Defect Detection in the Wood Industry

2.3.1 Common Wood Defects

When considering wood as a material, wood defects encompass various irregular texture structures and harm inflicted by factors either during the growth phase of the wood or during its processing (Ding et al. 2020). Wood displays a high degree of diversity. The characteristics of the material differ among trees of the same type, different types of trees, and among different sections of a single tree (Ross 2015). In particular, genetics, wind, and weather contribute to a broad range of variations. These factors shape the properties of wood, resulting in significant diversity of the material (Ross et al. 1998). The definition of a defect is a property of the material that makes the material inappropriate for its intended purpose (Estévez et al. 2003). From an economic perspective, this refers to any characteristic of the material that diminishes its market value (Kollmann & Côté 1968).

In the case of structural applications, e.g. in building, defects in the material significantly reduce its utilization (Koman et al. 2013). This is due to the fact that defects may affect the mechanical properties of the material and thus reduce it's required flexibility or rigidity (Ji et al. 2019, Kamal et al. 2017, Cao et al. 2016, Koman et al. 2013). When undetected, these issues may even result in structural collapse in severe cases, e.g. when used in construction (Zielińska & Rucka 2021).

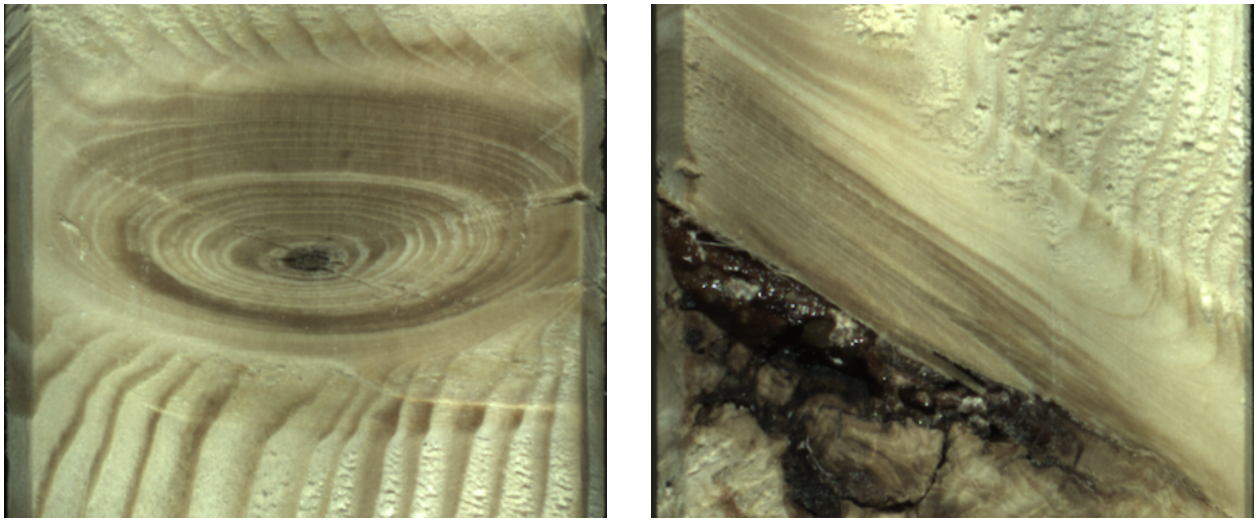
Besides safety concerns, customers of primary or secondary wood products expect products free from defects that could affect their visual appearance or structural integrity (Ding et al. 2020). This concern regarding defects extends even to wood surfaces that are not directly visible to the customer. Numerous defects lead to unsatisfactory outcomes post-painting, consequently impacting the overall appearance of the end product (Qayyum et al. 2016). The presence of defects, whether within the material itself or in the final product, results in a reduction in value and the potential for customer dissatisfaction (Abdullah et al. 2020). Consequently, to ensure the expected quality of a product, a thorough defect detection process becomes necessary (Rahiddin et al. 2020, Ji et al. 2019). However, even though defects often render the material unusable for many purposes, they may be desirable in other cases, for example due to aesthetic reasons (Kollmann & Côté 1968). But even in this case, the detection of the abnormality plays an important role during the manufacturing process.

Knots

Wood defects come in many different shapes and forms. According to Kollmann & Côté (1968), the by far most prevalent type of defects are knots. Knots form when a branch grows around the base of a tree stem, becoming incorporated into it. The cambium layers of both the stem and the branch remain connected as long as the branch is not deceased. If the branch deceases, the continuous connection between the cambium layers is broken. This forms loose or encased knots. Knots can take on different shapes in the cut pattern, depending on the direction in which they are cut. Figure 2.1 showcases this behavior. Knots affect the mechanical characteristics of the material because the fibers are interrupted. This plays an important role especially in structural applications (Kollmann & Côté 1968). Due to the changes in the surface, dead knots or cracked knots in particular can also lead to defects in surface processing, e.g. by painting. Loosened dead knots can also create holes or gaps in the in material.

Cracks

Cracks (see figure 2.2 (a)) and splits create gaps between the fibers and can emerge from changes in temperature such as drying stresses or frost, or from mechanical forces applied to the material. Cracks negatively impact the mechanical properties of wood and reduce its rigidity (Li et al. 2021, Kollmann & Côté 1968). Cracks also affect the quality of the paint finish and can lead to unsightly results that do not stand up to quality standards. Particularly problematic are fine cracks that are not detected during quality control and continue to expand as the coating dries in a heat chamber.



(a) (b)

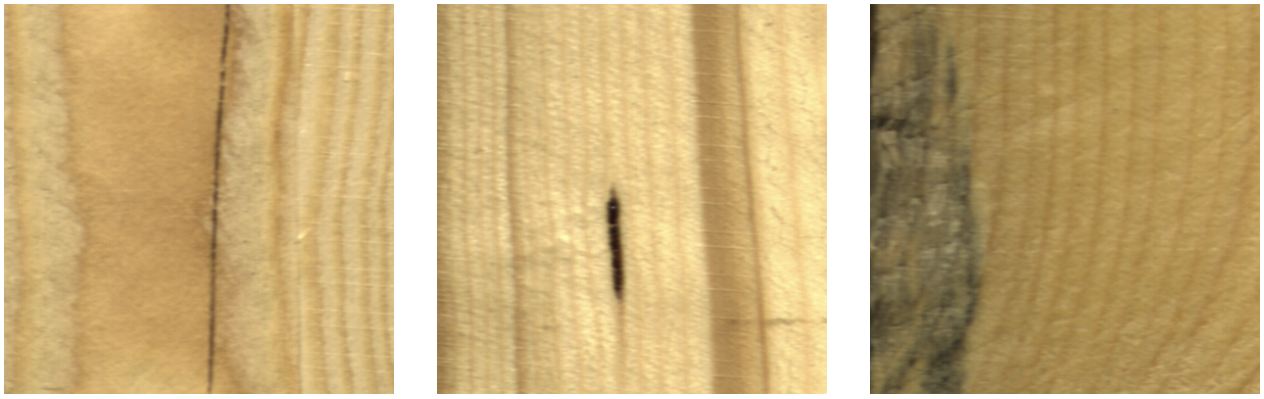
Figure 2.1: Appearance of knot defects in sawn lumber. Figure (a) displays a live knot after a transverse section through the branch. Figure (b) displays a deceased knot after a cut along the axis of the branch.

Resin Pockets

Resin pockets or pitch pockets are holes in the material filled with resin (see figure 2.2 (b) for reference). These may arise from cracks of uncertain origin occurring at the cambium layer. Insects might also be the cause of damages that subsequently lead to resin pickets. The formed holes or cracks are then filled with resin (Kollmann & Côté 1968).

Blue Stain and Fungal Defects

Blue stain is a wood defect caused by wood-decaying fungi of types ascomycota and deuteromycota (Broda 2020). The spores of these fungi are transported by insects to different hosts. A high moisture content and a lack of host defense mechanisms favor the organism (Uzunovic et al. 2008). As a result, infestations with fungi can happen when the material is stored under suboptimal conditions. Blue stain affects sapwood and causes blue or grey discolorations of the material (see figure 2.2 (c) for reference). The phenomenon results into increased water permeability due to a deterioration of the pit membranes. Furthermore, the defect can impact the material's aesthetic appeal and market value (Broda 2020). Infestations by certain species may affect the mechanical properties of wood by compromising the integrity of cell walls (Uzunovic et al. 2008). Other types of defects caused by fungi include brown-rot, white-rot, soft-rot, and mould (Broda 2020).



(a) (b) (c)

Figure 2.2: Wood defects in sawn lumber. Figure (a) displays a dark crack. Figure (b) displays a resin pocket. Figure (c) displays blue stain. Images taken from (Kodytek Pavel et al. 2021, Kodytek et al. 2022)

Mineral Streaks

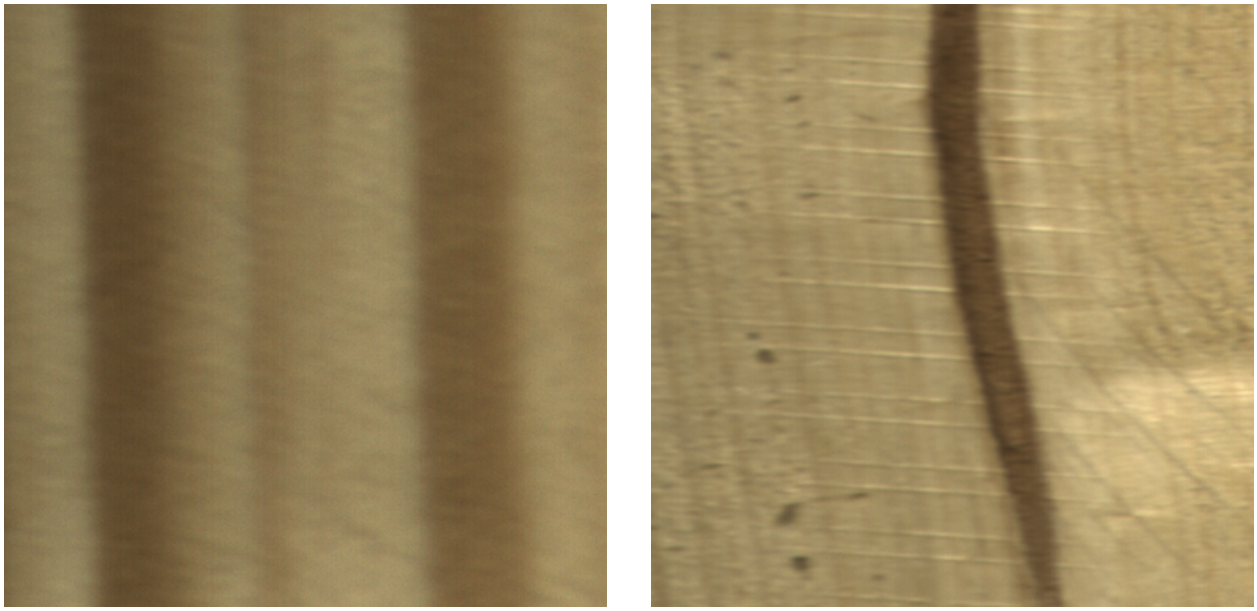
Mineral streak or mineral discolouration is a defect that manifests itself as dark discoloration along the grain pattern (see figure 2.3 (a) for reference). Such discolorations are caused by accumulations of various minerals in the tissue (Kollmann & Côté 1968). Mineral streak occurs in trees grown from mineral-rich soil. In addition to optical factors, the mineral streak defect can play a role in the further processing of the material, as certain minerals may lead to increased wear of cutting tools (Uzunovic et al. 2008).

Pith

Pith, as shown in figure 2.3 (b), refers to the central tissue within a tree stem. After its formation, his typically softer material does not grow as the tree matures (Akachuku & Abolarin 1989). Pith is classified as a grading defect (Gazo et al. 2020).

2.3.2 Traditional Defect Inspection Process

Defect inspection plays a vital role in ensuring the quality assurance of wood manufacturing processes (Zhang et al. 2015). Although visual evaluation is widely applied for quality control in the wood industry (Ross et al. 1998), the inhomogeneity of wood in terms of texture and external appearance makes it difficult to detect surface defects (Cao et al. 2016). Nondestructive defect detection procedures are required to meet requirements of utilization rate and waste reduction (Yang et al. 2020). This is especially important, as the material is the primary cost component



(a) (b)

Figure 2.3: Darkish mineral streak (a) and dark brown pith (b) defects on wood surfaces. Images taken from (Kodytek Pavel et al. 2021, Kodytek et al. 2022)

and represents a major lever for cost reduction. Producers therefore strive to enhance their yield, which denotes the quotient of produced lumber surface and processed lumber surface (Buehlmann & Thomas 2002). Buehlmann & Thomas (2002) further put the share of lumber costs at 12 % to 15 % of the total manufacturing cost, while using furniture manufacturers as an example. The defect detection process in the wood industry is mostly performed manually by inspection by experienced operators (Gao et al. 2022, Hwang et al. 2021). Defects are then highlighted and the affected material undergoes rework or is scrapped (Tu et al. 2021). According to Urbonas et al. (2019), the process of manual inspection rarely exceeds an accuracy of 70 %. This leads to various potential problems. Manual inspection is a repetitive, tedious and time-consuming process that introduces subjectivity and can lead to a number of human errors (Ericsson et al. 2021, Urbonas et al. 2019, Ding et al. 2020). The decisions made by workers in the forest products industry have a direct impact on the quality of the resulting wood products, and the utilization of the material (Conners et al. 1997).

A study by Buehlmann & Thomas (2002) on manual defect detection by operators demonstrated that over 78 % of the operator's decisions were not optimal. In more than 43 % of cases, defects were not recognized as such (Buehlmann & Thomas 2002). These inaccuracies in determining defects and their respective locations lower the amount of usable material, and thus, increase the cost of manufacturing (Buehlmann & Thomas 2002). In addition to being error prone, the manual inspection process involves additional manpower, resulting in increased labor costs (Ke

et al. 2016). Moreover, the demand for well-trained operators for the inspection task exceeds the market supply (Kryl et al. 2020). The results are not consistent with current industrial standards for efficiency and accuracy (Li et al. 2021). According to Hashim et al. (2015), 22 % of the material that is discarded incorrectly is due to human error. Ultimately, manual inspection processes increase the cost of the product (Buehlmann & Thomas 2002). Therefore, a fast, accurate defect detection process is necessary to realize revenue increases (Shi et al. 2020).

2.3.3 Nondestructive Evaluation and Testing

Nondestructive evaluation (NDE) refers to a scientific approach used to evaluate the properties and quality of a material without damaging it, and thus, without compromising its intended functionality (Ross et al. 1998, Sun 2022). The methods used within this field to collect accurate data concerning the material are referred to as nondestructive testing (NDT) technologies. The via NDT obtained information is then used to determine how to use the material (Ross 2015). Depending on the objective of the test procedure as well as the properties of the material to be examined, many different test procedures can be used (Arriaga et al. 2023).

In the wood specific case, NDT methods are employed to assess potential issues of wood without the need for destructive sampling or cutting (Sun 2022). According to Zielińska & Rucka (2021), NDT technologies are utilized for wood to detect defects including cracks, decay, failures, and deformations which could compromise the material's usability. Additionally, nondestructive testing methods are critical in the evaluation of important parameters of the material such as its MOE, MOR, moisture content, density, stiffness and inhomogeneity (Zielińska & Rucka 2021). The fast and accurate identification of targeted wood defect information serves as a foundation for the automated detection of wood defects. Automated approaches can bring significant economic benefits to enterprises by optimizing the utilization of wood raw materials and improving overall operational efficiency in the wood industry (Sun 2022).

2.3.4 Automated Approaches

Automated NDT approaches for a variety of purposes have been applied in the wood industry. Amongst others, these purposes include species identification (Hwang & Sugiyama 2021), wood polish classification (Lin & Sanjaya 2021), detection of pith and annual rings (Habite et al. 2021, 2022), and quality measurement of timber bundles (Carratu et al. 2021).

The research in automated visual approaches for the detection of wood defects have raised increasing attention, particularly in countries with abundant wood resources, such as Scandinavia (Gu et al. 2009). According to Conners et al. (1997), machine vision technology has been leading

to improvements in efficiency in the industry since the early 1980s. In the beginning, this was mostly for determining the dimensions of the material and not for determining characteristics of the material. Since then, significant research efforts have been dedicated to developing alternative technologies aimed at optimizing the value of the final products (Conners et al. 1997).

Chapter 3

Theoretical Background

This chapter introduces relevant concepts that are important for the understanding of the problem and the approach of this work. The chapter provides an overview of deep learning, computer vision (specifically image classification and object detection), and transfer learning. This is important within the context of defect detection in wood manufacturing. First, foundational concepts of deep learning are explained. This includes deep feedforward neural networks, activation functions, and model training procedures. Second, convolutional neural networks are detailed. Third, it is elaborated upon how these techniques and algorithms are used in image classification and object detection tasks. Finally, the concept of transfer learning is introduced, which allows to leverage pre-trained models for improved defect detection with limited or no labeled data. This chapter establishes the necessary theoretical foundation for the subsequent empirical investigation in wood manufacturing defect detection.

3.1 Deep Learning

The following description and notations are based on Goodfellow et al. (2016). Deep feedforward networks, or similarly, deep neural networks, feedforward neural networks, multilayer perceptrons (MLP) are fundamental models in deep learning. The models are called neural networks because they are an extremely simplified representation of biological neural networks and connect multiple features in the structure of a directed acyclic graph (DAG). The functions are connected in chains one after the other, resulting in a structure of multiple layers. The number of these layers is referred to as the depth of the network, from which the term deep learning is derived.

Deep feedforward networks serve as the foundation for many practical applications, includ-

ing object detection and natural language processing (NLP). Their purpose is to approximate a function f^*, which varies by task. For instance, in the case of a classifier network, $f^*(x)$ maps an input x to a class y. The naming of a feedforward network comes from the fact that information flows through the network from input x, through intermediate computations, to output y without any feedback connections (Goodfellow et al. 2016). Each deep neural network has one input layer, one output layer and an arbitrary number of hidden layers. Figure 3.1 demonstrates the flow of information through a feedforward network.

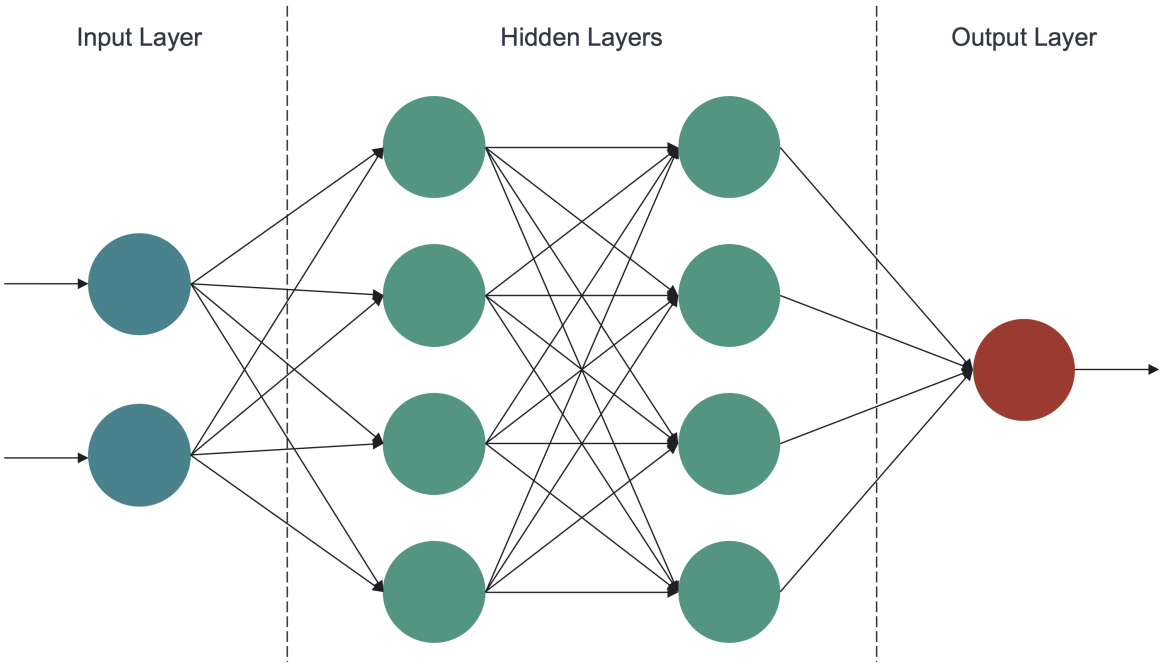


Figure 3.1: Structure of a feedforward network. This figure visually explains the composition of a feedforward neural network, featuring two hidden layers between the input and output layers. The information flows from left to right, without any lateral or feedback connections.

During the training of a neural network, the objective is to make the predicted output $f(x)$ match the desired output $f^*(x)$. The training data consists of examples of $f^*(x)$ at different input points. These examples are typically noisy and approximate. Each training example, denoted as x, is associated with a corresponding label y, which is an approximation of $f^*(x)$. The training examples explicitly, which enter the network via the input layer, define the expected behavior of the output layer, which is to produce a value close to y for each input x. However, the behavior of the intermediate layers, known as hidden layers, is not directly specified by the training data. The learning algorithm must determine how to utilize these hidden layers in order to achieve the desired output function. Therefore, the hidden layers are responsible for capturing and representing the complex relationships between the input and output (Goodfellow et al. 2016).

The hidden layers consist of several neurons connected in parallel, and can be seen as individual elements of a vector that the layer represents. The dimension of these vector-valued layers, i.e. the number of parallel neurons, is called the width of the neural network (Goodfellow et al. 2016). Each neuron has multiple input connections and one output connection. Normally, the neurons of one layer are connected to all neurons of the previous as well as the following layer. The output h of a hidden layer is computed by $h = g(Wx + c)$, where $g(z)$ is an activation function, W is a weight matrix, x is a vector of input features, and c is a bias vector. Wx denotes a matrix multiplication. The output h is composed of all the outputs of the individual neurons in that layer. The output of a single neuron j is computed by $h_j = g(\sum_i^n x_{ij} w_{ij} + c_j)$. A structural representation of a neuron can be seen in Figure 3.2.

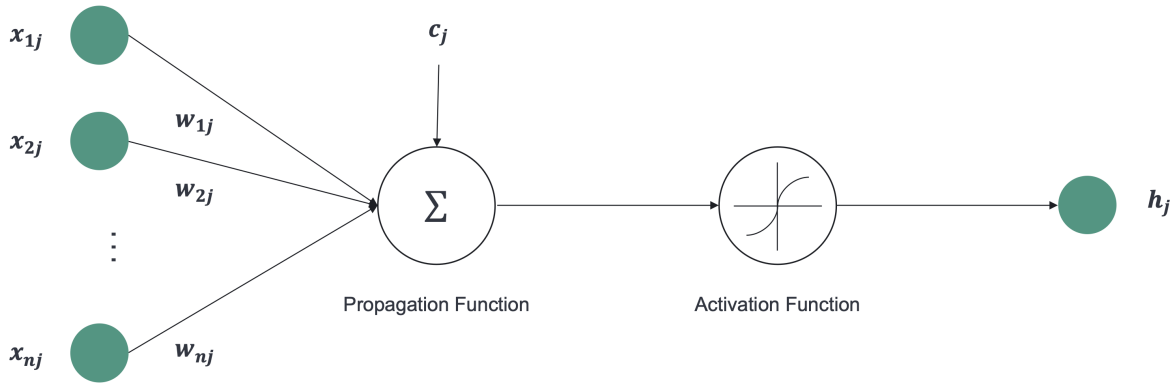


Figure 3.2: A single hidden layer neuron j with multiple inputs x_i, associated weights w_i and bias c. The output h is computed via passing the propagation function Σ in the activation function.

The use of activation functions is inspired by the action potential in biological neurons. Activation functions are utilized to introduce non-linearity in a mathematical process by performing element-wise operations to the input after it has undergone linear transformation (Misra 2019). Activation functions are essential components in neural networks as they prevent the multilayer network from collapsing into linear models. In the discipline of deep learning, different types of activation functions are commonly used. Examples of activation functions are the logistic sigmoid, the hyperbolic tangent, the rectified linear unit (ReLU) (Nair & Hinton 2010), and newer variants such as Swish (Ramachandran et al. 2017), and Mish (Misra 2019). Table 3.1 provides an overview of the activation functions mentioned.

Figure 3.3 provides visual depictions of the activation functions and corresponding derivatives introduced in table 3.1. For a more comprehensive exploration of activation functions, interested readers are encouraged to refer to the referenced material.

Table 3.1: Common activation functions in deep learning and their derivatives.

Activation function	$f(x)$	$\frac{df}{dx}$
Sigmoid (σ)	$\frac{1}{1+e^{-x}}$	$f(x)(1-f(x))$
Hyberb. tangent	$\tanh(x)$	$1-f(x)^2$
ReLU	$\max(0,x)$	$\begin{cases} 0, & \text{if } x<0 \\ 1, & \text{if } x>0 \\ \text{undefined}, & \text{if } x=0 \end{cases}$
Swish	$x\cdot\sigma(\beta x)$	$\beta f(x)+\sigma(\beta x)(1-\beta f(x))$
Mish	$x\cdot\tanh(\text{softplus}(x))$	$\text{sech}^2(\text{softplus}(x))\cdot x\cdot\sigma(x))+\frac{f(x)}{x}$

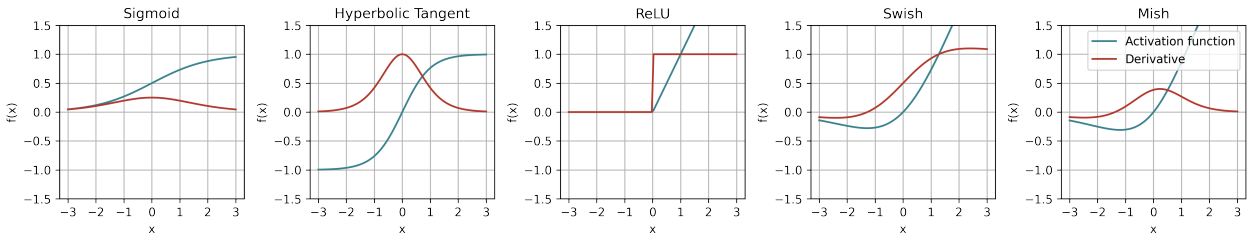


Figure 3.3: Graphic representation of the activation functions and their derivatives.

Training and Optimization

During training, the network with a mapping $y = f(x;\theta)$ learns the optimal parameter values $\theta = \{W^{(1)},\dots,W^{(l)},b^{(1)},\dots,b^{(l)}\}$, where W and b denote the weights and biases of the layers, respectively. This has the aim to approximate f to f^* as close as possible, and is achieved by minimizing a loss function $J(\theta)$. A common algorithm for adjusting θ is back-propagation with gradient descent (Rumelhart et al. 1986, Lecun et al. 1998). The steps of the algorithm are:

1. Forward-pass: The input values x are passed forward through the layers of network to obtain a prediction y. The weights of the network are randomly initialized. Random weight initialization is necessary to break the symmetry between neurons. If neurons with the same inputs had the same initial weights, they would be updated in the exact same manner during training and thus fail to learn distinct patterns (Goodfellow et al. 2016). Depending on the purpose and architecture, popular initialization techniques such as Xavier initialization (Glorot & Bengio 2010) or He initialization (He et al. 2015) might be used.

2. Loss calculation: A loss function $J(\theta)$ is used to determines the loss of the network which is to be minimized. This loss function is a measure of the dissimilarity between the predictions made by a model and the actual target values (He et al. 2020). The choice of the loss functions varies depending on the task and the nature of the target values.

3. Backward-pass: The algorithm starts from the output layer and successively calculates the gradient of the loss function with respect to each weight and bias in the network, i.e. to obtain $\frac{\partial J(\theta)}{\partial W^{(l)}}$, and $\frac{\partial J(\theta)}{\partial b^{(l)}}$, respectively. This recursive implementation of the chain rule of calculus is computationally efficient (LeCun et al. 2015). The gradient indicates the direction and magnitude of the weight adjustment needed to minimize the loss.

4. Update weights and biases: The calculated gradients are used to update the weights and biases of the network. The update is performed iteratively using the gradient descent algorithm, which adjusts the weights in the opposite direction of the steepest ascend of the loss function, according to $\theta_{k+1} = \theta_k - \eta\nabla_\theta J(\theta_k)$. Here, η is the learning rate that determines the step size in the weight update process. Choosing an appropriate learning rate is an important factor as the parameter influences the effectiveness and speed of the training process.

5. Steps 1. - 4. are repeated until a pre-defined termination criterion is reached.

Common issues that can occur during backpropagation are vanishing and exploding gradients. Vanishing gradients refer to the situation where the gradients of the loss function become extremely small as they are backpropagated through the network layers. This can happen in deep

networks when the gradients are multiplied by small values repeatedly. Especially when using sigmoid activation functions in combination with random weight initialization, the gradients can disappear due to the nature of the activation function (Glorot & Bengio 2010). As a result, the weights of the earlier layers are updated very slowly. This leads to slower convergence and difficulty in learning meaningful representations (Bengio et al. 1994, Glorot & Bengio 2010). Exploding gradients refer to the situation where the gradients become extremely large during backpropagation. This can happen when the gradients are multiplied by large values repeatedly. This may cause the weights to update dramatically and lead to unstable training (Pascanu et al. 2013). Special initializations of the weights and the insertion of normalization layers in the architecture can counteract this phenomenon (He et al. 2016*a*).

3.1.1 Convolutional Neural Networks

This section aims to provide an overview of convolutional neural networks, including their architecture. This is to establish a solid understanding of their functioning, which is essential for comprehending the subsequent discussions and analyses related to wood defect detection.

Convolutional neural networks (CNN), or convolutional networks, ConvNets, were introduced by LeCun et al. (1989) in 1989 and have since been applied to various domains. Like deep feedforward networks, CNNs are inspired from biology. The overall design shares similarities with the hierarchical organization observed in the visual cortex, a region of the brain (Fan et al. 2021, LeCun et al. 1989). CNNs have proven to be highly effective in the field of computer vision since the early 2000s. They have been widely applied with great success for tasks such as object detection, region segmentation, and image recognition (LeCun et al. 2015). However, the availability of large databases with millions of labeled images such as ImageNet (Deng et al. 2009), as well as the widespread availability of powerful hardware such as GPUs, have led to increased interest in the architecture and to great advances and successes in the field of visual recognition (Simonyan & Zisserman 2014, Zeiler & Fergus 2014, Krizhevsky et al. 2012, LeCun et al. 2015). Since the breakthrough of the deep convolutional neural network AlexNet (Krizhevsky et al. 2012) at the ImageNet 2012 classification benchmark, the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), the models have steadily been evolved and outperformed older architectures. Popular architectures such as VGG (Simonyan & Zisserman 2014), Inception (Szegedy et al. 2015), ResNet (He et al. 2016*a*), EfficientNet (Tan & Le 2019), and ConvNeXt (Liu et al. 2022) introduced a variety of new features and design principles and have been widely used in the computer vision community.

The naming of CNNs results from the mathematical operation of convolution, which distinguishes the networks from conventional deep feedforward networks (Goodfellow et al. 2016).

Conventional deep feedforward networks are not suitable for processing large image data due to their architecture. An RGB image file with a resolution of 256×256 is used as an example. The values of the individual pixels are used as input features and the associated parameters are learned by the neurons. For an image file of this size, there would be $256 \times 256 \times 3 + 1 = \sim 196,000$ learnable parameters per neuron in the first hidden layer alone. For a layer with a width of 300 neurons, this results in about 59 million trainable parameters. Considering that deep neural networks typically have a large number of hidden layers, it becomes apparent why this is computationally infeasible. CNNs on the other hand incorporate so-called sparse interactions (Goodfellow et al. 2016). This refers to the use of kernels which are smaller than the input size. This approach allows for the detection of relevant features by analyzing only a subset of pixels within the input. As a result, CNNs have a lower amount of trainable parameters which leads to improved memory efficiency (Goodfellow et al. 2016). CNNs also benefit from weight sharing (or parameter sharing), i.e. applying the same kernel to multiple locations in the input. This property reduces memory requirements and increases training efficiency (Dumoulin & Visin 2016, Gu et al. 2018, Goodfellow et al. 2016). In addition, the convolution operations conducted in CNNs preserve the ordering of the data. Whereas deep feedforward networks flatten the input data to a vector before applying matrix multiplications, convolution performed on multidimensional arrays preserves the structural information of the data. Hence, CNNs are valuable for solving tasks that require understanding spatial or temporal relationships (Dumoulin & Visin 2016).

CNN Architecture

CNNs are composed of multiple stages, with the front stages each consisting of stacks of multiple convolutional layers. These convolutional layers are equiped with activation functions, critical for introducing non-linearity between them (LeCun et al. 2015). Within these layers, the feature maps are computed through the utilization of multiple convolutional kernels (Gu et al. 2018). Multiple kernels form a convolutional filter. Between the stacks of convolutional layers there are (optionally) pooling layers, which reduce the resolution without having trainable parameters. Subsequent to this, one or more fully-connected layers follow, each featuring nonlinear activations, leading to the classification head (Loukadakis et al. 2018, LeCun et al. 2015, Simonyan & Zisserman 2014, Zeiler & Fergus 2014). To provide visual context, figure 3.4 visually outlines the architecture of VGG-16 (Simonyan & Zisserman 2014), a popular CNN model that won the ILSVRC in 2014.

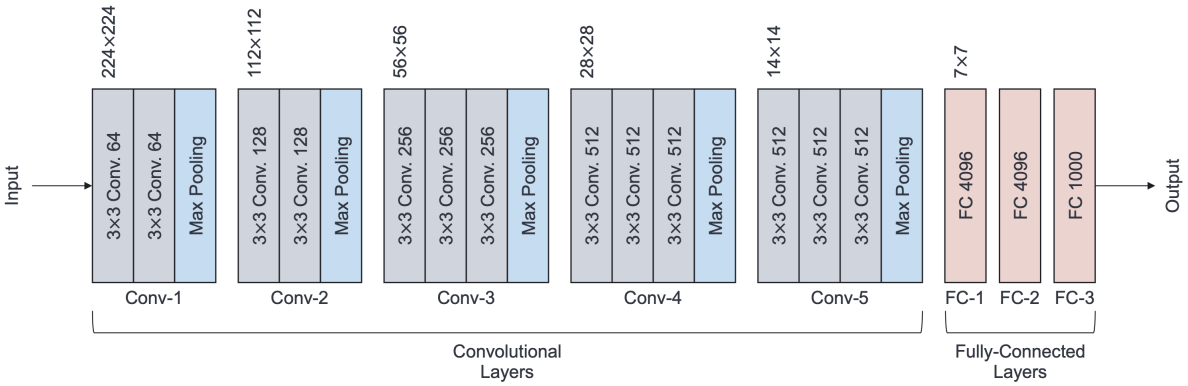


Figure 3.4: Visual representation of a CNN architecture using the example of VGG-16. The architecture consists of combined stages of convolutional and pooling layers, as well as fully-connected layers. The final fully-connected layer is followed by a softmax function for classification. The resolution of the input is denoted on top of each block. The pooling layers each reduce the height and width of the feature map by half.

Convolution Operation

CNNs work with multidimensional arrays as input, so-called tensors. A tensor refers to a collection of numerical values organized in a structured pattern on a grid-like structure, which can have a varying number of dimensions (Goodfellow et al. 2016). Matrices can therefore be considered as two-dimensional tensors, while vectors are one-dimensional tensors. An RGB image as input can be represented as a three-dimensional tensor with dimensions corresponding to its height, width, and color channels. In contrast to RGB images with three channels, grayscale images have only one channel and can thus be represented as two-dimensional tensors. In common implementations, four-dimensional tensors are used as input values, where the fourth dimension denotes the batch size (Goodfellow et al. 2016).

The inputs of convolutional layers are called input feature maps. Similarly, the output of convolutional layers is called the output feature map (Dumoulin & Visin 2016). Each neuron within a feature map is associated with a distinct region of adjacent neurons in the preceding layer. This group of neighboring neurons is known as the receptive field (Gu et al. 2018). The fundamental operation in CNNs is convolution. The output feature map is generated by applying a sliding window called a kernel to the input feature map and performing element-wise multiplication followed by summation, and the addition of a trainable bias. The convolved output is then process by an activation function (Gu et al. 2018). Mathematically, the convolution performed in CNNs (denoted by the asterisk $*$) can be written as:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

where $S(i,j)$ describes the output value at position (i,j), $K(m,n)$ refers to the value of a 2D kernel at position (m,n), and $I(i+m,j+n)$ to a 2D input image at position $(i+m,j+n)$ (Goodfellow et al. 2016). For multiple input feature maps, it is necessary to use a three-dimensional filter, i.e. a specific kernel for each input feature map. The final output feature map is then computed by element-wise summation of the convolved feature maps (Dumoulin & Visin 2016). Figure 3.5 demonstrates convolution with a three-dimensional filter.

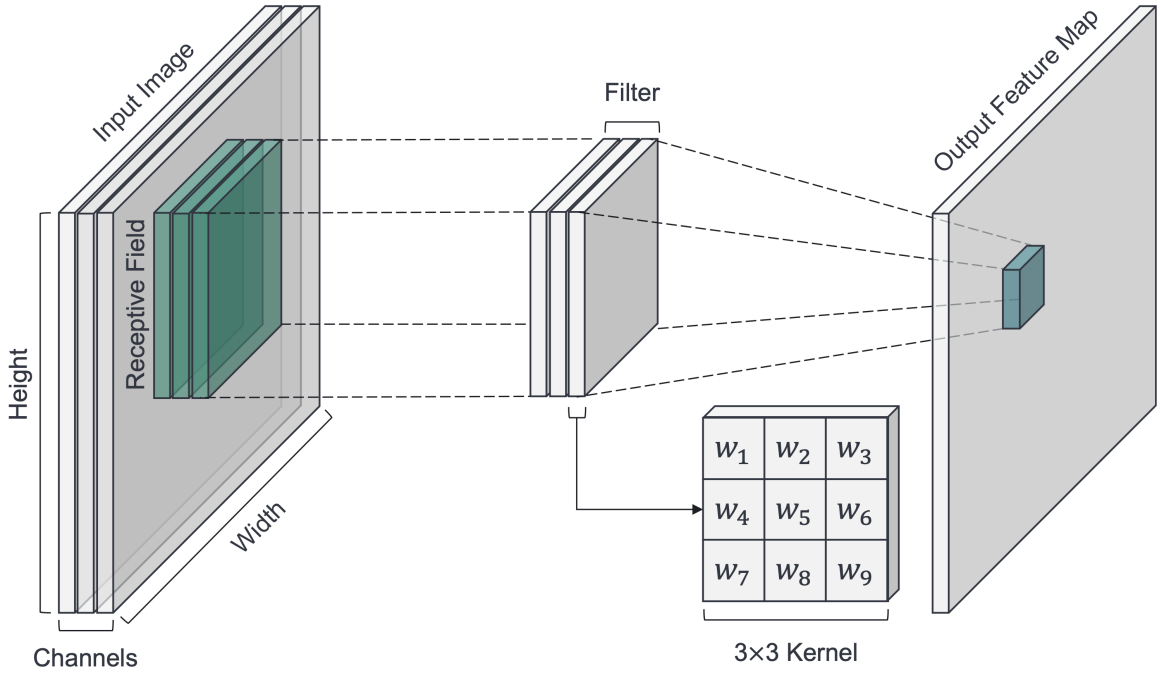


Figure 3.5: Demonstration of the convolution operation on an RGB input image. A 3×3 kernel with trainable weights is used. The size of the receptive field determines how much information of the original image is stored in one feature map pixel. To keep the input resolution, padding can be used.

Pooling

Pooling layers may be added in order to reduce the resolution of feature maps by aggregation pixel values (Krizhevsky et al. 2012). Pooling aggregates an area $A = p \times p$ of the feature map, where A represents an area of multiple pixels p. There are several variants of pooling, such as max-pooling $max(a \in A)$ or average pooling $\frac{1}{|A|}\sum_{a \in A} a$. Reducing the resolution of the feature maps reduces the amount of data in the network while preserving the information of the dominant features, even if the input is subject to small translations (Goodfellow et al. 2016). The latter property is also referred to as translation invariance. This is a useful property, e.g. in the case of defect detection on wood surfaces. Here, the pixel-precise position of the defect usually

does not matter, whereas the mere presence of the defect does.

Regularization Techniques

To prevent overfitting and increase training stability of CNNs, regularization techniques are commonly implemented.

Batch normalization (Ioffe & Szegedy 2015) is used to counteract the constantly changing distributions of the layer inputs (internal covariate shift) during training which results from the parameter adjustment of the previous layers. Batch normalization normalizes the activations of each neuron in a neural network layer to have zero mean and unit variance using the statistics computed from a mini-batch of samples (Ioffe & Szegedy 2015). Moreover, batch normalization helps smoothing the landscape of the underlying optimization problem (Santurkar et al. 2018). As a result, this technique can improve training stability and speed up convergence, while also reducing the sensitivity to the choice of initialization and learning rate (Ioffe & Szegedy 2015, Li et al. 2019).

DropoutHinton et al. (2012), Srivastava et al. (2014) involves randomly zeroing out the outputs of neurons during each training iteration with a specified probability p. During training, these neurons then are not considered during the forward and backward pass. During testing, all neurons are active but the weights previously deactivated neurons are scaled down. (Srivastava et al. 2014, Li et al. 2019) Dropout introduces noise in the network, which helps to prevent the network from relying on specific neurons (Krizhevsky et al. 2012, Hinton et al. 2012). As a result, the network must increase its robustness (Krizhevsky et al. 2012).

L1 and L2 regularization (weight decay) (Krogh & Hertz 1991) are regularization techniques to prevent overfitting and improve the generalization ability of the model. These add a penalty term to the loss function $J(\theta)$, scaled by a parameter α (or λ). The L1 regularization term $\sum_{i=1}^{n} |\theta_i|$ partially sets weights to zero and therefore leads to sparse features. L1 regularization is therefore used for feature selection rather than for pure avoidance of overfitting. The penalty term of L2 regularization $\sum_{i=1}^{n} \theta_i^2$ encourages the model's weights to be small, but not necessarily zero. This aims to prevent the model from relying too heavily on any particular feature (Goodfellow et al. 2016).

Data augmentation is another variant to avoid overfitting and make the model more robust. Data augmentation techniques artificially increase the size of the training dataset to provide more information to the network (Shorten & Khoshgoftaar 2019). Data augmentation can be achieved by geometric transformations, such as translation, rotation, zoom, and mirroring, or by color changes, such as changes in brightness, contrast, etc. of the input data. Training

data can also be generated synthetically, e.g. by using Generative Adversarial Networks (GANs) (Goodfellow et al. 2014, 2020).

Fully Convolutional Networks

Fully convolutional networks (FCN) were introduced by Long et al. (2015) in 2015, and are primarily designed for dense pixel-level predictions such as semantic segmentation. The architecture is used in object detection algorithms such as R-FCN (Dai et al. 2016), YOLO (Redmon & Farhadi 2018), FCOS (Tian et al. 2019), and SSD (Liu et al. 2015). FCNs consist entirely of convolutional layers by exchanging any fully connected layers with convolutions. As a result, in contrast to CNNs, FCNs work with inputs of arbitrary size, and process the entire image at once during learning and inference (see figure 3.6 for reference). The output of the network retains the same spatial dimensions as the input (Long et al. 2015). FCN use additional so-called *deconvolutional* layers that perform upsampling operations and thus boost the resolution of the output. A skip architecture is then used to combine the unrefined information from deeper layers with refined information from shallow layers. This enables the architecture to predict finer details (Ronneberger et al. 2015, Long et al. 2015).

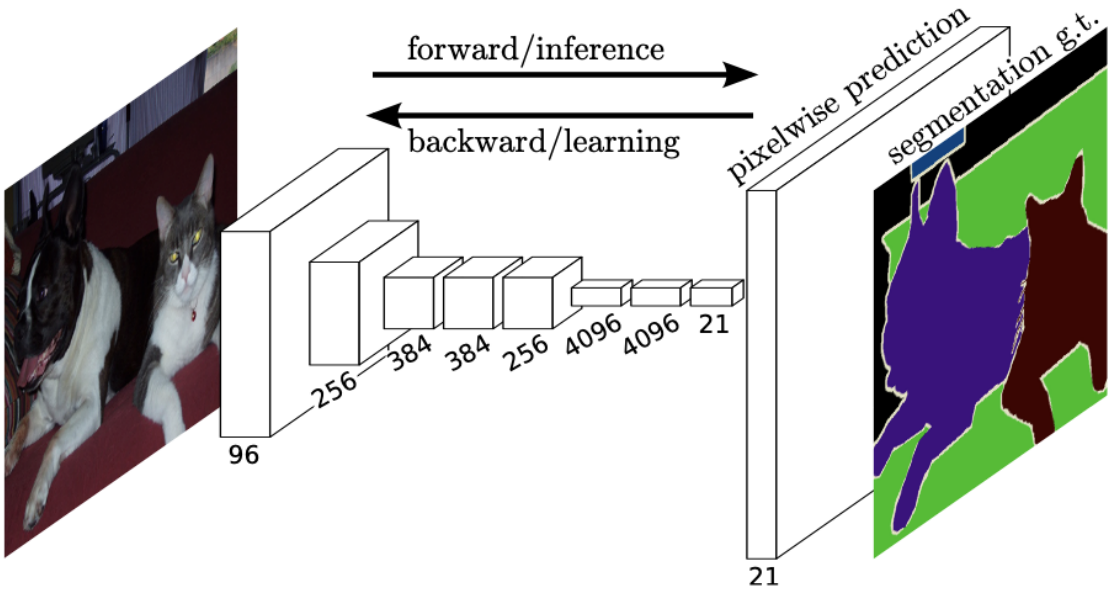


Figure 3.6: High-level Architecture of fully convolutional network (FCN). No fully connected layers are used, hence learning and inference happen on image level. Figure taken directly from (Long et al. 2015).

3.2 Computer Vision

Computer vision deals with replicating the property of vision of humans and animals. The discipline of computer vision has made great strides in its research in the recent past and is one of the main application areas for advanced machine learning and deep learning technologies (Goodfellow et al. 2016). Computer vision is applied in numerous domains for a wide variety of purposes and is used by humans and machines for decision making. Computer vision applications play a vital role in quality inspection, essential for achieving the objectives of zero defect manufacturing (Psarommatis et al. 2019).

In this section, two fundamental tasks of computer vision are explained: image classification and object detection. Image classification helps classify image data into different classes. Object detection goes one step further by identifying the exact location of the object in addition to classification. Both technologies form the basis for various applications in modern technology. Due to the increasing efficiency, accuracy and availability of the algorithms, these technologies are becoming more and more attractive, even for smaller companies such as in the wood processing sector.

3.2.1 Image Classification

Image classification is one of the fundamental disciplines in computer vision. Image classification treats the input image as a whole and aims at assigning a predefined label to the image, indicating the associated object class (Wang & Su 2019). An indication of the presence or absence of objects in an image is achieved via binary labels in image classification (Lin et al. 2014). With the advancements in deep learning, Convolutional Neural Networks (CNNs) have become the state-of-the-art approach for image classification tasks.

Binary Classification

Binary classification is a task in image classification where the objective is to classify the image into one of two mutually exclusive classes. This typically means determining whether the image belongs to a specific class or not, e.g. *defect*, or *no defect* in the case of wood defect classification. Here, the two classes are typically divided into a positive and a negative class or binary coded with 0 and 1.

Multi-Class Classification

Multi-class classification aims to classify an image into one of several mutually exclusive classes. The input image is assigned exactly one label that corresponds to the most appropriate class among multiple options. In contrast to binary classification, the number of classes is greater than two. This task is commonly encountered in scenarios where images can belong to different categories, such as different wood species.

For multi-class classfification, a softmax function is usually used in the final layer of the CNN. The softmax function converts the activated output of the previous layer into a normalized probability distribution over all potential classes. Each class is assigned a probability, and the class with the highest probability determines the prediction. The softmax function for a problem with K different classes, and an input vector $z \in \mathbb{R}^K$ is defined as:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j^K e^{z_j}} \quad \text{for } i = 1, \ldots, K$$

Multi-Label Classification

In contrast to multi-class classification, multi-label classification assigns zero or more labels to the input at the same time, where each label describes a specific class in a set of given classes. Figure 3.7 demonstrates the difference. Here the classes are not mutually exclusive. Multi-label classification is of practical relevance, since images of real-world scenarios often contain different objects at the same time (Wei et al. 2016). For example, a wooden surface may have defects of different categories at the same time. Using multi-label classification algorithms, the input can then be classified not only as defective or not defective, but also based on the type of defects present. This can be useful, for example, if only a subset of the possible defects is considered relevant in the production context and qualifies the product for rework or scrap, for example.

For multi-label classification, where each input can belong to multiple classes simultaneously, using the softmax function is not appropriate, as all class probabilities are summed to 1. Instead, the sigmoid function can be used in order to allow each class to be evaluated separately. A threshold is then applied to determine whether a class is present or absent based on its probability. The sigmoid (σ) function is defined as:

$$\sigma(z_j) = \frac{1}{1 + e^{-z_j}}$$

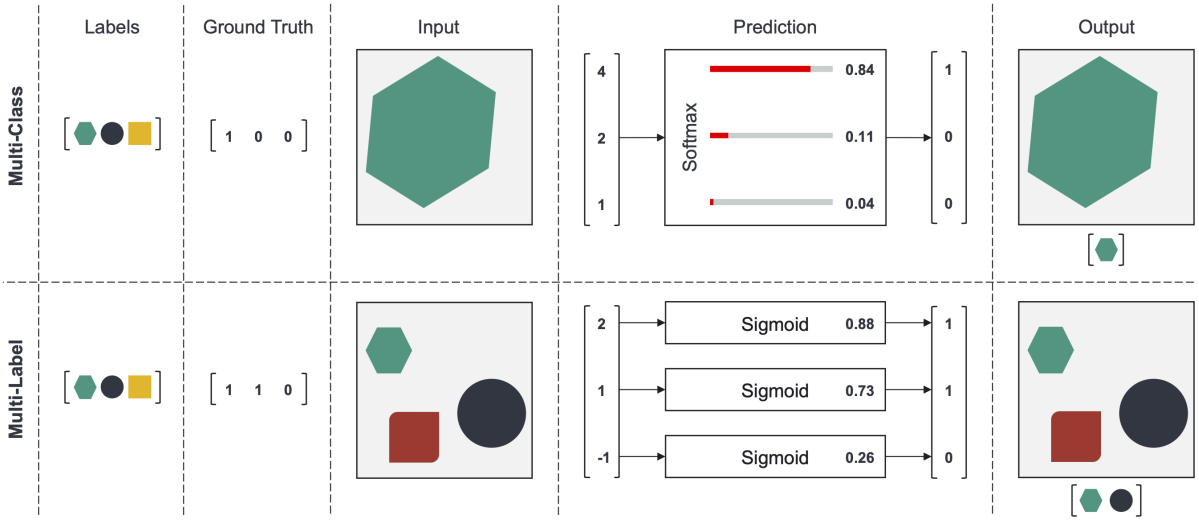


Figure 3.7: Visual representation of the difference between multi-class classification (top), and multi-label classification (bottom). Multi-class problems use the softmax function to assign a specific label. The sigmoid function used for multi-label classification problems can assign multiple labels to the input.

3.2.2 Object Detection

Object detection is a computer vision task that involves identifying and localizing multiple objects of interest within an image or video. The localization of objects distinguishes object detection from image classification, where objects are merely classified. Object detection can be used, for example, to detect and localize defects on wooden surfaces, e.g. if certain regions of the surface are of particular relevance to the absence of defects, or if the location of the defect influences further processing steps. The goal is to classify present objects of interest but also locate their positions via bounding boxes (Lin et al. 2014, Carion et al. 2020).

There are different architectures used for object detection. The classic object recognition algorithm uses a sliding window approach, where a window of defined size passes through different scales of the input step by step. A classification network then classifies the respective image regions (LeCun et al. 1989, Vaillant 1994, Lin, Goyal, Girshick, He & Dollar 2017). However, the advent of deep learning brought significant improvements in object detection performance in terms of accuracy and speed (Lin, Goyal, Girshick, He & Dollar 2017). This led to the development of new dominant architectures, namely, two-stage object detectors such as R-CNN (Girshick et al. 2014), Fast R-CNN (Girshick 2015), Faster R-CNN (Ren et al. 2015), and R-FCN (Dai et al. 2016), one-stage detectors such as Overfeat (Sermanet et al. 2013), SSD (Liu et al. 2015), RetinaNet (Lin, Goyal, Girshick, He & Dollar 2017), FCOS (Tian et al. 2019) and YOLO (Redmon et al. 2016). More recently, transformer architectures, which represent the state of the art in NLP,

where adapted to object detection tasks (Dosovitskiy et al. 2020, Liu et al. 2021, Zhang & Yang 2021, Fan et al. 2021, Carion et al. 2020). The latter architecture is out of scope of this work, and the interested reader is referred to the literature.

A basic component of object recognition algorithms are bounding boxes. Bounding boxes are minimal two-dimensional boxes that surround the object and thus represent the location and extend of the object (Zhou et al. 2019). A bounding box is defined by four parameters and associated with a class label. Bounding boxes can be encoded in different formats such as pixel coordinates or normalized coordinates relative to the image size. Typical representations of bounding boxes are *Center XYWH*, *XYWH*, and *XYXY*, where the center coordinates, width and height, the top left coordinates, width and height, and the top left and bottom left coordinates are specified, respectively.

The architecture of CNN based models typically consists of two components, a backbone and a head. The backbone is used for feature extraction and is usually a CNN (for example ResNet or VGG) that has often been pre-trained with data from one of the large image databases. The head performs the actual prediction of the object class and its localization based on the extracted features (Terven & Cordova-Esparza 2023, Bochkovskiy et al. 2020). The differences between two-stage and one-stage detectors are manifested in the structure of the head (Bochkovskiy et al. 2020).

Two Stage Detectors

As the name indicates, two stage detectors follow a two-step process for object detection. In the first stage they produce a limited set of potential object locations, which is denoted as region proposal. In the second stage, these locations are individually classified using a CNN, determining whether they belong to foreground classes or the background (Lin, Goyal, Girshick, He & Dollar 2017, Carranza-García et al. 2020).

A region proposal refers to a candidate bounding box in an image that is likely to contain an object of interest. In object detection tasks, region proposals are generated to identify potential object locations before further processing and classification. The purpose of generating region proposals is to reduce the computational burden by focusing only on regions that are more likely to contain objects, rather than processing the entire image. By generating a set of region proposals, the object detection algorithm can narrow down the search space and concentrate its efforts on these regions, improving efficiency and reducing computational resources. The original R-CNN and Fast R-CNN use a selective search algorithm for the task of region proposal, which functions as modules independent of the detection network. The difference between the former and the latter is that R-CNN forward passes each region proposal through the CNN, whereas the

CNN in Fast R-CNN processes the set of proposed regions and the image in one forward pass (Girshick et al. 2014, Girshick 2015, Ren et al. 2015). Later architectures such as Faster R-CNN and R-FCN do not rely on selective search, and instead, use so-called region proposal networks (RPN). RPN are end-to-end trainable fully convolutional networks (FCN) (Long et al. 2015) that take the image feature maps as input. The network uses a sliding window approach to go over the feature map and generates a number of so-called *anchors* (or anchor boxes, prior boxes) at each position, translation invariant rectangular boxes with predefined sizes. These anchors are then classified and regressed via two separate layers in order to determine whether the region is an object or background, and predict the offsets between anchors of foreground objects and their ground-truth boxes (Ren et al. 2015, Huang et al. 2017, Tian et al. 2019).

Two stage detectors are highly accurate but suffer from excessive computational requirements. Complex pipelines with individually trainable components lead to elaborate training processes. This property makes them unsuitable for embedded or mobile systems. Furthermore, even with high-end hardware, their processing speed falls short for real-time applications (Redmon et al. 2016). For reference, Faster R-CNN, which is the fastest and most accurate two stage detector, operates at a only 7 FPS according to (Liu et al. 2015, Redmon et al. 2016). For real-time object detection in fast-paced production environment, two-stage detectors are too slow.

One Stage Detectors

In contrast to two-stage detectors, one stage detectors predict bounding box coordinates and class probabilities in one pass (Redmon et al. 2016, Liu et al. 2015, Sermanet et al. 2013). The detectors typically consist of a single neural network. This simplifies the detection process by directly predicting object locations and categories without explicit region proposal steps, resulting in speed gains at the potential cost of detection accuracy. This design enables one-stage detectors to operate in real-time applications, as they are computationally efficient.

Newer algorithms often extend the models by additional layers between feature extractor and head. In version 4 (Bochkovskiy et al. 2020) and later iterations of YOLO, these intermediate layers or networks are referred to as neck. The neck operates with feature outputs from different layers of the backbone as input (Bochkovskiy et al. 2020). Different outputs from shallow and deep layers in the form of feature maps can be combined via this intermediate component and thus contribute to the extraction of information about different scales(Terven & Cordova-Esparza 2023). This is often achieved via feature pyramid networks (Lin, Dollar, Girshick, He, Hariharan & Belongie 2017). These have bottom-up and top-down pathways that use lateral connections to fuse feature maps from different stages of the backbone with highly sampled outputs from deeper layers (Lin, Dollar, Girshick, He, Hariharan & Belongie 2017). Figure 3.8

showcases the high-level backbone-neck-head architecture of modern detectors.

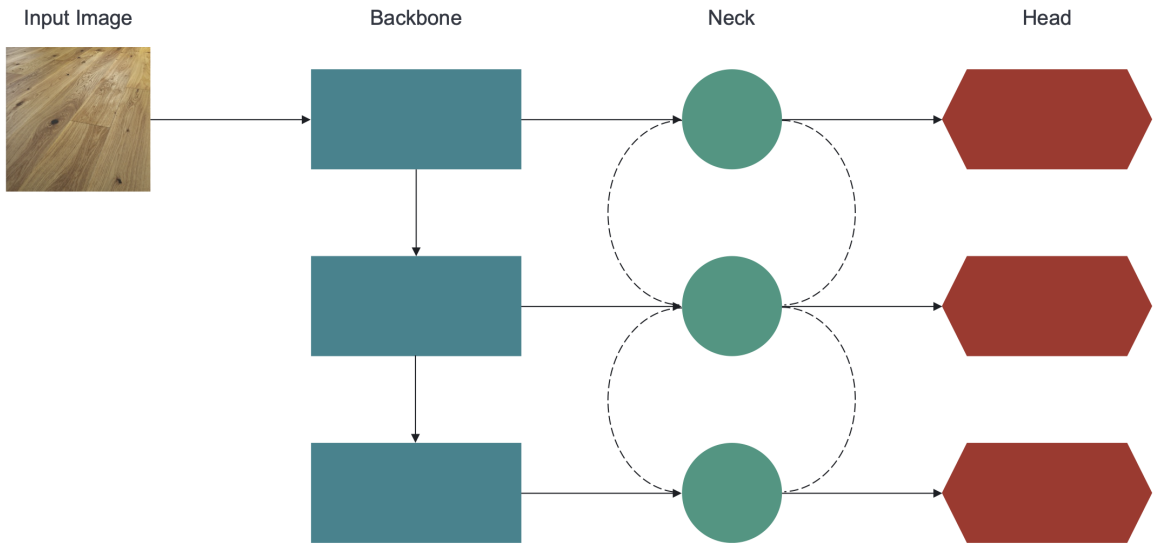


Figure 3.8: High level architecture of modern one-stage detectors with backbone, neck, and head. The backbone works as a feature extractor, the neck fuses features from different stages via lateral connections, and the head performs classification and regression operations. Figure adapted from (Terven & Cordova-Esparza 2023).

For common object detectors, a distinction can be made between anchor-based and anchor-free approaches. Anchor-based approaches use, as with modern two-stage detectors, predefined anchors in different scales which are subsequently classified as object or background and refined by regression in their offset to the ground-truth box (Tian et al. 2019). Algorithms using this anchor-based approach then typically aim to minimize a joint loss function that integrates both classification and regression objectives (Huang et al. 2017). However, anchor-based approaches have some drawbacks such as domain specificity and increased complexity due to the need for hyper-parameters and more expensive computations. This complexity can impact the inference time and system efficiency (Tian et al. 2019, Ge et al. 2021, Zhou et al. 2019). Anchor-free approaches predict bounding boxes based on the position of the object. Here, center-based and keypoint-based approaches are used, which define bounding boxes relative to the center coordinates or region or to particular coordinates, such as extreme points of objects, respectively (Zhang et al. 2020). By eliminating various hyper-parameters and computations, anchor-free approaches can significantly reduce the complexity of training (Ge et al. 2021).

One factor for the loss of accuracy of the original one-stage detectors is a high foreground-background class imbalance (Oksuz et al. 2021). These object detectors propose high amounts of regions, of which the main part belongs to the background class. Due to the high proportion of negative class, the training efficiency is reduced. In addition, the class imbalance has a

negative impact on the training process by overpowering the model and might lead to the development of less effective models that perform poorly in accurately detecting objects (Carranza-García et al. 2020, Lin, Goyal, Girshick, He & Dollar 2017). The developers of RetinaNet (Lin, Goyal, Girshick, He & Dollar 2017) address this issue by reshaping the loss function. This approach is called focal loss and counteracts the high influence on the gradients of easy negatives, i.e. image regions that are strongly separated from the object classes by their features. This is implemented by assigning low importance to the easy negatives within the loss function. The model thus learns primarily from difficult examples, which ultimately leads to improved detection performance (Lin, Goyal, Girshick, He & Dollar 2017). Other approaches have addressed the issue of class imbalance by improving their architectures, and implementing data augmentation (Redmon et al. 2016, Bochkovskiy et al. 2020). By adding new or modified examples to the training data, the diversity of objects appearances can be increased and thus lead to a more balanced training (Oksuz et al. 2021). The benefit of this approach is that it does not affect inference time (Bochkovskiy et al. 2020). Hard negative mining, as used for example in (Liu et al. 2015), uses only the negative examples resulting from the standard boxes with the highest confidence loss, so that the ratio between negative and positive classes is reduced.

3.3 Model Selection

In this section, we highlight the models used for the analysis. The model architectures and special features are discussed, which are important for the selection of the models with respect to the use case. Furthermore, their advantages over other models are explained. The section is divided into model selection for image classification tasks and model selection for object detection.

3.3.1 Models for Image Classification

ResNet

ResNet was developed and introduced by He et al. (2016*a*) of Microsoft Research in 2015. With this model, the authors placed first in the 2015 ILSVRC classification challenge as well as other tasks in the competition. The name ResNet is short for residual network, which refers to the introduction of residual connections, a specific type of shortcut connections (see figure 3.10). Residual connections allow for the propagation of information from earlier layers directly to later layers by identity mapping. This is to solve the problem of degrading training performance and higher error of networks that are increased in depth. As a result, this architectural design

facilitates the training of deeper networks with improved accuracy, and might mitigate the vanishing gradient problem (He et al. 2016*a*).

He et al. define $\mathcal{H}(\mathrm{x})$ as a mapping with inputs x, which is desired to be approximated by several successive layers. In the case of residual learning, the layers now learn a residual mapping $\mathcal{F}(\mathrm{x}) := \mathcal{H}(\mathrm{x}) - x$ instead of $\mathcal{H}(\mathrm{x})$. This residual mapping is the difference between the desired output and the input. This allows the original function $\mathcal{H}(\mathrm{x})$ to be reformulated as $\mathcal{F}(\mathrm{x}) + \mathrm{x}$. This technique aims to simplify the learning process of deep networks by allowing the network to focus on learning the residuals, rather than the entire mapping from input to output (He et al. 2016*a*).

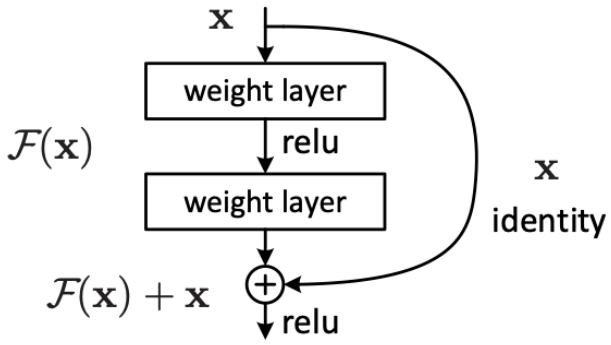


Figure 3.9: A building block of ResNet with a residual connection. Shortcut connections allow information to skip layers of the CNN without adding computational complexity. Figure taken from (He et al. 2016*a*).

The ResNet architectures are designed with a modular approach where building blocks of the same shape are stacked together (He et al. 2016*b*). He et al. (2016*b*) refer to these structures as *residual units*. The original ResNet architecture comes in different depths (18, 34, 50, 101, and 152 layers). For the deeper ResNets with 50 or more layers, the so-called bottleneck architecture is used. Here, blocks of three layers are used for the residual functions instead of blocks of two layers as used in ResNet-18 and ResNet-34. The name of the bottleneck blocks results from the design with a 3×3 convolutional layer between two 1×1 convolutional layers (in contrast to two 3×3 convolutional layers in shallower ResNets). With the first 1×1 layer the input is compressed so that the 3×3 layer has less computational overhead. With the second 1×1 layer the resolution is increased again. Here, the advantage of the shortcut connections with regard to time complexity becomes particularly apparent (He et al. 2016*a*).

In 2016*b*, the authors of ResNet introduced an updated version of ResNet with improvements regarding to the shortcut connections. Furthermore, residual units were pre-activated with batch normalization in order to reduce overfitting and enhance training performance (He et al. 2016*b*).

For this analysis, ResNet-18 and ResNet-50V2 are used.

EfficientNet

The family of deep learning architectures EfficientNet was introduced by Tan & Le (2019) of Google AI in 2019. The authors address the problem of scaling CNNs, which in previous research was often accomplished by scaling one of the three parameters: depth, width, and resolution. EfficientNet, on the other hand, scales a baseline model (EfficientNet-B0) by applying a compound scaling factor ϕ to all of the three dimensions simultaneously. Figure X visualizes the types of scaling of CNNs.

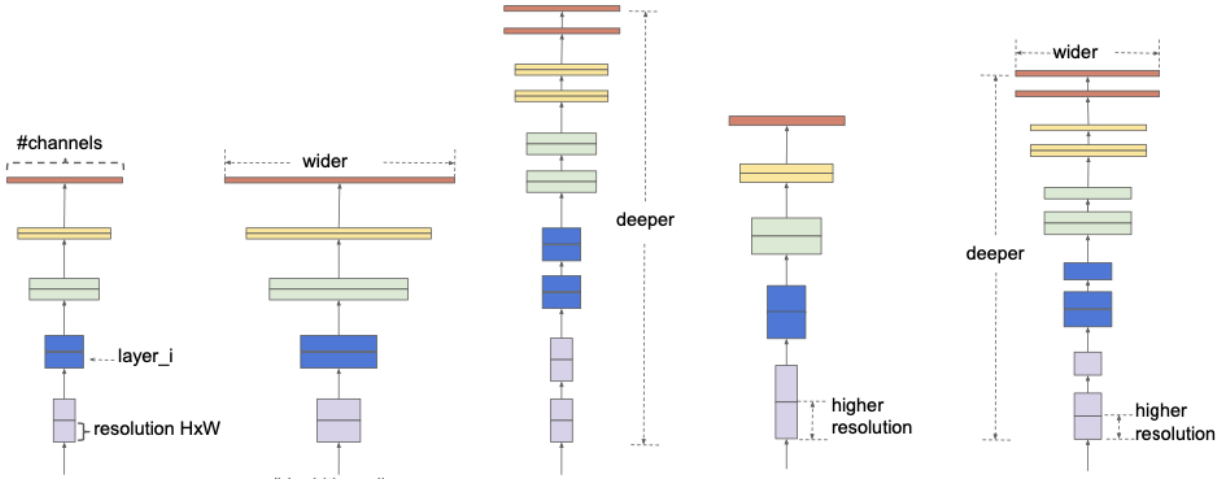


Figure 3.10: Visual representation showcasing different scaling methods of CNN models. From left to right: (1) baseline model, (2) width scaling - increasing the width of the network, (3) depth scaling - increasing the depth of the network, (4) resolution scaling - changing the input resolution of the network, and (5) compound scaling - simultaneous scaling of widht, depth, and resolution using a compound coefficient ϕ. Figure taken from (Tan & Le 2019).

The parameters width, depth and resolution were determined by the scaled constants α^{ϕ}, β^{ϕ}, and γ^{ϕ}, which are subject to the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$. The constants were determined via grid-search in first place. For benchmarking against alternative algorithms, the ImageNet Challenge (Russakovsky et al. 2015) was used. According to the authors' benchmarks, the baseline model achieves accuracy comparable to ResNet-50, but has only 5.3 million trainable parameters compared to 26 million in the case of ResNet-50. In the paper, further upscaled EfficientNets, called EfficientNet-B1 to EfficientNet-B7 were presented. These demonstrated better performance with increasing scaling at the cost of higher training time. However, with a maximum of 66 million trainable parameters, they remain relatively small compared to other architectures in the same performance category. Also in the case of transfer learning, less complex EfficientNet models showed comparable performance to known architectures.

In 2021, EfficientNetV2, an extension of the original models, were proposed by Tan & Le. The

family of seven new models focus on lower complexity and improved training speed. This is achieved by architectural changes in the convolutional layers, and improvements in the area of compound scaling. Adaptive regularization, which varies in strength depending on image size, is also introduced. The adaptations have improved the performance while reducing the training speed Tan & Le. In this thesis, the B0 version of the EfficientNetV2 extenstion is compared against the ResNet models.

3.3.2 Models for Object Detection

YOLOv7

YOLOv7 was introduced in 2022 by Wang et al. (2023) and is one of the latest iterations of the YOLO family of one-stage detectors. The model was published by the authors of YOLOv4 (Bochkovskiy et al. 2020) and YOLOR (Wang et al. 2021). Besides changes in architecture, the variant introduced a range of optimization techniques focused on an improved accuracy when detecting objects in images without negatively affecting inference speed. These methods, named *bag-of-freebies* were introduced in the YOLOv4 iteration and only affect the training of the model in terms of its required time (Bochkovskiy et al. 2020, Wang et al. 2023, Terven & Cordova-Esparza 2023).

Wang et al. (2023) made the algorithm available in different scales in order to be suitable for a wide range of hardware including mobile devices. Compared to YOLOv4, YOLOv7 models exhibit a reduction of approximately 40 % in trainable parameters and 20 % in computations. Versions for edge computing reduce computation nearly by half. Various changes to the algorithm achieve this reduction in complexity while simultaneously improving the performance of the model. A compound method comparable to that of EfficientNet is used for scaling (Wang et al. 2023). YOLOv7 uses the design strategy extended efficient layer aggregation network (E-ELAN), which should lead to efficiency gains (Wang, Liao & Yeh 2022). This technique helps controlling the longest shortest gradient path in order to ensure that gradients can flow effectively through the layers during training. The method uses different operations on features to enhance the network's learning capacity (Wang et al. 2023, Terven & Cordova-Esparza 2023). Additionally, the architecture introduces a modified re-parametrized convolution operation, changes in the integration of batch normalization, and an additional *auxillary head* that helps training the network. The swish activation function is used in variants of the algorithm. Binary cross-entropy loss and focal loss (Lin, Goyal, Girshick, He & Dollar 2017) are implemented as loss functions (Wang et al. 2023).

YOLOv8

YOLOv8, like YOLOv5, was developed by Jocher et al. (2023), CEO of Ultralytics, and released in 2023. The developers are not part of the actual authors of the YOLO algorithm and have not yet published a formal research paper on the algorithm. The algorithm has been published on GitHub (Jocher et al. 2023) and the associated documentation can also be found on the company website. As with other iterations of YOLO, the algorithm is available in five differently scaled versions (nano, small, medium, large, and extra large).

According to Terven & Cordova-Esparza (2023), YOLOv8 adopts a decoupled head structure, enabling separate processing of objectness, classification, and regression tasks. The separate processing of the individual tasks leads to improvements in the performance of the model. For the bounding box regression, Distribution Focal Loss (DFL) (Li et al. 2020) and Distance-IoU (D-IoU) (Zheng et al. 2020) loss are implemented as loss functions (Terven & Cordova-Esparza 2023). DFL is an extension of Focal Loss, which leads to improved estimates of the boxes by formulating their positions as general distributions (Li et al. 2020). D-IoU loss enhances training performance by integrating the standardized distance between the prediction and ground truth in the function (Zheng et al. 2020). According to Terven & Cordova-Esparza, this choice of loss functions leads to improvements in detection, especially when objects of smaller dimensions are to be detected. The mish activation function is used in variants of the algorithm.

3.4 Transfer Learning

While traditional machine learning technology has achieved considerable success and has been successfully used in numerous practical applications, it has reached its limits in certain real-world scenarios (Zhuang et al. 2021). The conventional approach to supervised machine learning relies on an isolated learning procedure, i.e., training a specific learning engine on a labeled data set intended for that purpose (Ruder et al. 2019). Both data sets should share the same feature space, and the distribution of the training data set should match the distribution of the target data (Day & Khoshgoftaar 2017, Pan & Yang 2010). If this is not the case, this can lead to a deterioration of the prediction ability of a model on unknown instances (Weiss et al. 2016). In many real-world scenarios, however, this is not feasible. In reality, the training data and the test data may differ in this respect, for example, they may have different dimensions in the feature space, or they may have different variables (Day & Khoshgoftaar 2017).

In order for the model to generalize as well as possible to unknown data, large amounts of labeled training data are necessary (Zhuang et al. 2021, Sharma et al. 2019). Deep learning in particular heavily relies on extensive training data. This high dependence on data is attributed to

the need for a substantial volume of examples to uncover and comprehend the underlying patterns within the data (Tan et al. 2018). However, labeled data are generally difficult to generate and often costly (Tan et al. 2018). Manual annotation of data performed by experts is time consuming and prone to human error. Furthermore, within the industrial context, classified data or data containing personal information cannot be used for such purposes or shared within the organization due to the inherent security risks associated with data transfer (Wang, Zhou, Liang, Yan & She 2022).

The concept of transfer learning stems from the basic idea that people can apply existing knowledge to obtain better solutions to novel problems, or solve them in a shorter time (Pan & Yang 2010). In machine learning, the paradigm of transfer learning is to transfer knowledge from one source domain $\mathcal{D}_S$ to another domain, the target domain $\mathcal{D}_T$ in order to predict unknown instances (Gao et al. 2019, Zhuang et al. 2021, Neyshabur et al. 2020, Wang, Zhou, Liang, Yan & She 2022, Ruder et al. 2019). During transfer learning, a network is pre-trained using a dataset and adapts its weights to the generic data. These weights are then used in subsequent tasks to initialize the network. This initialization enables the network to solve these new tasks with fewer data (Kolesnikov et al. 2020). This is especially useful when the target domain has non-sufficient amounts of data or no labeled data at all (Day & Khoshgoftaar 2017). Furthermore, the initialization reduces computational demands, making the process more efficient and cost-effective than training from scratch (Kolesnikov et al. 2020, Neyshabur et al. 2020). Transfer learning has been applied for many purposes, including natural language processing (NLP) (Devlin et al. 2018, Ruder et al. 2019), object detection (Ren et al. 2015, Reis et al. 2023, Urbonas et al. 2019), image classification (Gao et al. 2022, Uzen et al. 2021), semantic segmentation (Long et al. 2015, He et al. 2017), and others.

Mathematically, a domain $\mathcal{D}$ can be represented as $\mathcal{D} = (\mathcal{X}, P(X))$, where $\mathcal{X}$ denotes the feature space and $P(X)$ represents the marginal probability distribution. The feature space $\mathcal{X}$ consists of a collection of feature vectors, typically denoted as $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$, which describe the attributes of a specific input instance within that domain (Day & Khoshgoftaar 2017, Pan & Yang 2010, Zhuang et al. 2021). These features can be quantitative, categorical, or a combination of both.

Pan & Yang (2010) further define the concept of a *task* for a given domain $\mathcal{D}$. A task consists of a label space $\mathcal{Y}$ and a prediction function $f(\cdot)$ that maps the input instances to the label space $\mathcal{Y}$. Hence, a *task* $\mathcal{T}$ can be represented as $\mathcal{T} = (\mathcal{Y}, f(\cdot))$, or similarly, $\mathcal{T} = (\mathcal{Y}, P(Y|X))$. The prediction function is learned during the training process by means of the training data of the associated domain. The labeled training data is in the form $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. Here, $x_i \in \mathcal{X}$ describes the feature vector of the input instance, and $y_i \in \mathcal{Y}$ the associated label. The label space describes the set of all possible labels of the problem (Pan & Yang 2010). For example, in the context of

a wood defect classification problem, the label space $\mathcal{Y}$ could consist of the set {Blue_Stain, Crack, Dead_Knot}.

In conventional machine learning problems, the source and target domains as well as the tasks of the respective domains do not differ from each other, i.e. ($\mathcal{D}_S = \mathcal{D}_T$), and ($\mathcal{T}_S = \mathcal{T}_T$). In transfer learning problems, however, the source and target domains are not identical, or the tasks of the domains differ, i.e ($\mathcal{D}_S \neq \mathcal{D}_T$), or ($\mathcal{T}_S \neq \mathcal{T}_T$), or both. From the previous definition of a domain, it is clear that two domains differ if the feature spaces of the two domains are not the same. Likewise, two domains differ if the marginal probability distributions of the data sets differ. Similarly, it follows from the definition of a task that the tasks of two domains differ if the label spaces differ, or if the conditional probability distributions differ. The objective of transfer learning is to improve the predictive function of the target domain given some knowledge from the source domain.

Homogeneous and Heterogeneous Transfer Learning

Day & Khoshgoftaar (2017) further divide transfer learning into two basic principles: homogeneous transfer learning and heterogeneous transfer learning. In the context of homogeneous transfer learning, also referred to as domain adaptation, the only discrepancy lies in the marginal distribution between the source domain and the target domain ($P(X_S) \neq P(X_T)$). Here, the feature spaces of the original and target domains are in the same dimension ($\dim(\mathcal{X}_S) = \dim(\mathcal{X}_T)$), and both domains possess identical independent variables as well as labels, i.e. ($\mathcal{X}_S = \mathcal{X}_T$), and ($\mathcal{Y}_S = \mathcal{Y}_T$). (Day & Khoshgoftaar 2017).

In heterogeneous transfer learning, the source and target domains may exhibit dissimilar feature spaces ($\mathcal{X}_S \neq \mathcal{X}_T$) which are typically also non-overlapping, and also the label space and the dimension of the feature space may differ, i.e. ($\mathcal{Y}_S \neq \mathcal{Y}_T$), and ($\dim(\mathcal{X}_S) \neq \dim(\mathcal{X}_T)$). This makes the task of transferring knowledge across domains more challenging. Here, not only the marginal distribution may differ ($P(X_S) \neq P(X_T)$), but also the conditional distribution of the features may differ between the domains $P(Y_S|X_S) \neq P(Y_T|X_T)$ (Gao et al. 2019, Day & Khoshgoftaar 2017). An example of heterogeneous transfer learning would be the adaptation of a concrete defect classification model with corresponding features and classes to a wood defect detection model with corresponding features and classes.

Other Categorization Criteria

A different categorization of transfer learning arises with respect to the given label information. Hereby, transfer learning can be divided into the categories inductive, transductive and unsu-

pervised transfer learning. In inductive transfer learning labels for source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$ are available. In transductive transfer learning, only labels for the source domain $\mathcal{D}_S$ are present. In unsupervised transfer learning the label information for both domains is missing (Zhuang et al. 2021).

Transfer Learning in Practice

Transfer learning is a commonly employed technique in deep learning, particularly when leveraging pre-trained models. The typical procedure involves several steps. Initially, the model is trained on the original domain or pre-trained weights from established deep learning libraries are utilized, which are often based on extensive image databases like ImageNet (Deng et al. 2009) or COCO (Lin et al. 2014). The following paragraph refers to a tutorial of the deep learning library Keras (Chollet 2020). Figure 3.11 illustrates this concept. The classification head of the network, originally designed for ImageNet's 1000 classes, is typically removed, and a new trainable layer, including a tailored classification layer for the target domain, is added. Subsequently, certain layers of the pre-trained network are frozen, preserving their weights and biases. This ensures that these parameters remain unchanged during training. The newly added layers are then trained using data specific to the target domain. Optionally, fine-tuning can be applied by unfreezing parts or the entire pre-trained network. The unfrozen layers are trained on the target data with a very low learning rate. Incrementally adapting the pre-trained features to the target domain's data can lead to further performance improvements. Additionally, to expedite training speed, the pre-trained network can serve as a separate feature extractor. Here, the training data from the target domain is passed through the pre-trained network, and the resulting output data (features) act as input for a new trainable network. This approach ensures that the training dataset only goes through the pre-trained network once, as opposed to the conventional practice of passing it through the network during every epoch (Chollet 2020).

Effects of Transfer Learning

Neyshabur et al. (2020) investigate what ultimately leads to the success of transfer learning in the domain of CNN. Feature reuse plays a crucial role in transfer learning. The greatest benefit was achieved when the original and target domains shared visual features. Raghu et al. (2019) indicate that the effective reuse of meaningful features is predominantly observed in the lower layers of the network. Nonetheless, according to Neyshabur et al. (2020), even when applying transfer learning from distant domains that do not share any visual features, the use of transfer learning can yield a noticeable improvement. This improvement cannot be explained by

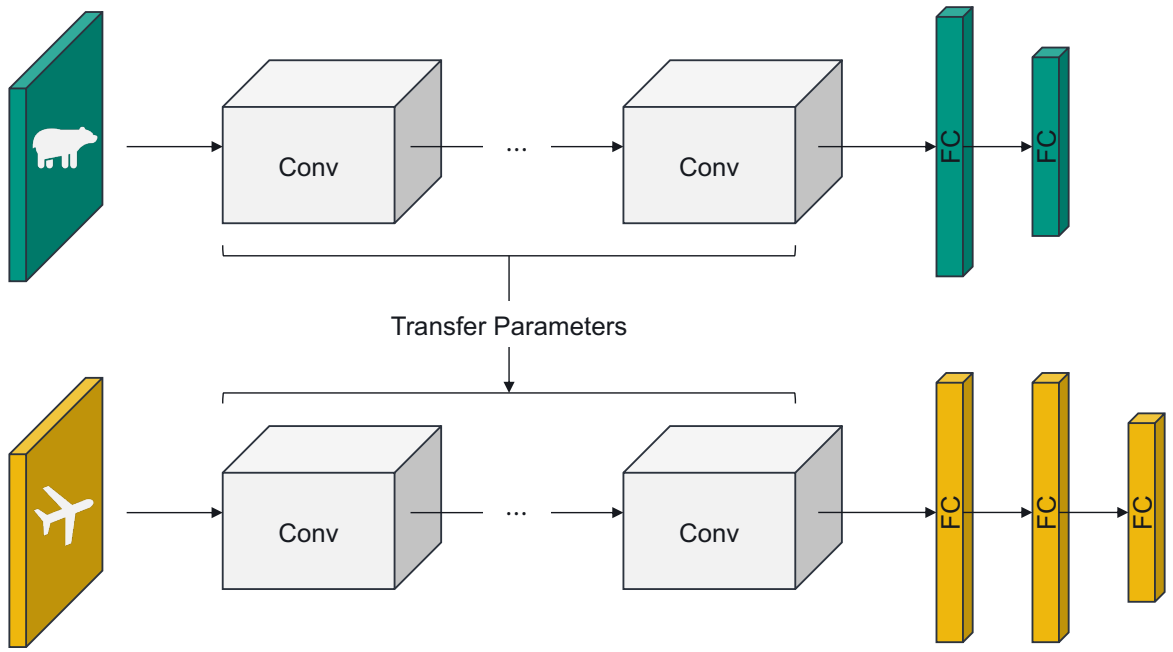


Figure 3.11: Illustration of transfer learning in CNNs. The figure showcases the transfer learning process, where a pre-trained CNN model on a large dataset is utilized as a starting point for a new domain.

feature reuse. The analysis shows that in this case the distribution of pixel values contribute to the benefits of transfer learning, especially in terms of optimization speed. In addition, it could be shown that the speed improvements are not dependent on the degree of feature reuse (Neyshabur et al. 2020). In the paper, the information the pixel value distribution inherits is referred to as low-level statistics (Neyshabur et al. 2020). In addition, the findings of Raghu et al. (2019) show that the feature-independent weight scaling improved by transfer learning improves the convergence speed (Raghu et al. 2019).

3.5 Performance Evaluation

Evaluation metrics serve as indicators utilized to evaluate the performance and overall effectiveness of machine learning models. These metrics objectively quantify the model's performance in terms of its quality, predictive capacity, and generalization ability, aiding in the comparison of various models. When choosing metrics a variety of factors has to be considered. These factors include the nature of the data, the type of task (e.g. regression or classification), and the intended goal. The metrics used in this analysis are defined in the following.

3.5.1 Evaluation Metrics for Image Classification

Recall and Precision

The following metrics use specific notations for classified instances. True positives (TP) represent the number of correctly predicted positive samples. False positives (FP) represent the number of negative samples incorrectly predicted as positive. False negatives represents the number of positive samples incorrectly predicted as negative.

Precision is a measure of the accuracy of positive predictions made by a model. It calculates the proportion of true positive predictions out of the total predicted positive instances, i.e. precision quantifies how reliable the positive predictions of a model are.

For a task that is not binary but comes with N classes, e.g. in a multi-label classification problem, macro precision (or macro-averaged precision) can be used. Macro precision calculates the precision value separately for each class i and then takes the average. Macro-averaged metrics give equal weight to each class. Precision and macro-precision are defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Macro Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i} = \frac{1}{N}\sum_{i=1}^{N}\text{Precision}_i$$

Recall measures the proportion of relevant instances correctly identified by the model out of all the actual positive instances. Recall captures the ability of the model to classify all the positive instances correctly, avoiding false negatives. For defect detection, a high recall value is important, since negatively classified defective workpieces that are not sorted out can lead to problems in later production steps. Similarly to macro precision, macro recall computes the arithmetic mean of all the per-class recall scores. Recall and macro recall are defined as:

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{Macro Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} = \frac{1}{N}\sum_{i=1}^{N}\text{Recall}_i$$

Recall and precision generally have an inverse relationship, i.e. increasing recall can lead to a decrease in precision and vice versa. To maximize recall, the model aims to detect as many

true positive instances as possible, e.g. by lowering the threshold for considering an object as positive. This typically leads to a higher number of false positives. To maximize precision, the model focuses on minimizing false positives by increasing the threshold for considering an object as positive. This may result in fewer detections overall, but with a higher confidence in their correctness.

In scenarios where precision and recall are both crucial factors to consider, the F1 score metric is often used. The F1 score denotes the harmonic mean of precision and recall. A high F1 score indicates a good balance between precision and recall. This means the model achieves both accurate positive predictions and captures a significant portion of positive instances. A low F1 score suggests an imbalance between precision and recall, indicating that the model may be biased towards either false positives or false negatives. The macro F1 score computes the arithmetic mean of all per-class F1 scores. F1 score and macro F1 score are defined as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Macro F1-Score} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} = \frac{1}{N} \sum_{i=1}^{N} \text{F1 Score}_i$$

3.5.2 Evaluation Metrics for Object Detection

Intersection over Union

Intersection over union (IoU) measures the overlap between the predicted bounding box and the ground truth bounding box, here denoted as B_p and B_{gt}, respectively. IoU is calculated by dividing the intersection area between the predicted and ground truth regions by the union area of the two regions (see Figure 3.12 for reference. IoU ranges from 0 to 1, where 1 indicates a perfect match between the predicted and ground truth regions, and 0 indicates no overlap. Formally, according to (Everingham et al. 2009), IoU can be expressed as:

$$\text{IoU} = \frac{Area(B_p \cap B_{gt})}{Area(B_p \cup B_{gt})} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

Typically, a threshold of $IoU > 0.5$ is set, for considering a prediction as correct (TP). Likewise an $IoU \leq 0.5$ is considered as an incorrect prediction (FP) (Everingham et al. 2009). A ground truth box that goes undetected corresponds to a false negative. It should be noted that true negatives are not used in the area of object detection, since this would include all possible correctly

undetected bounding boxes.

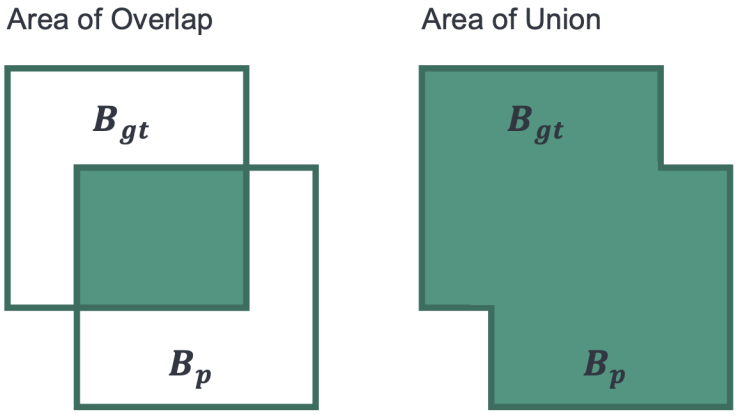


Figure 3.12: Illustration of the area of intersection and area of union between a ground truth and predicted bounding box.

Precision and Recall

In object detection, precision and recall depend on defined threshold values. Here, according to (Russakovsky et al. 2015), confidence scores s_{ij} are output for each prediction of an object location. This is done for all available classes, each with a total of N ground truth boxes. For each detection j in image $i \in I$, a binary variable z_{ij} is assigned with a value of 1 if the detection matches a ground truth box based on a threshold criterion t, and 0 otherwise. Hence, (Russakovsky et al. 2015) define precision and recall as:

$$\text{Precision}(t) = \frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{\sum_{ij} 1[s_{ij} \geq t]}$$

$$\text{Recall}(t) = \frac{\sum_{ij} 1[s_{ij} \geq t] z_{ij}}{N}$$

Average Precision and Mean Average Precision

Contrary to what the name suggests, the average precision metric does not simply describe the average of the precision value. Instead, the area under the precision-recall curve (see figure 3.13) is evaluated at different points. For the Pascal VOC challenge, an interpolated AP score was introduced (Everingham et al. 2009). This metric has since been used in many modern algorithms. The recall values between 0 and 1 are divided into 11 intervals r of the same size.

The precision values $p(\tilde{r})$ at a specific recall interval $\tilde{r}$ are interpolated by taking the maximum precision at each recall level, as follows:

$$p_{interp}(\tilde{r}) = \max_{\tilde{r}:\tilde{r}\geq r}(p(\tilde{r}))$$

The interpolation serves to smoothen the curve and to weaken effects of fluctuations. AP is then defined as the average of the interpolated precision values over all 11 intervals:

$$\text{AP} = \frac{1}{11} \sum_{r\in\{0,0.1,\ldots,1\}} p_{interp}(\tilde{r})$$

Mean average precision (mAP) averages AP over all classes and is typically evaluated at different IoU thresholds. For the MS COCO object detection challange, thresholds in the range of $[0.5, 0.95]$ in steps of 0.05 are used (Lin & Dollar 2016). The mAP score is defined as:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_c$$

Where C is the number of classes, and AP_c is the average precision of class c.

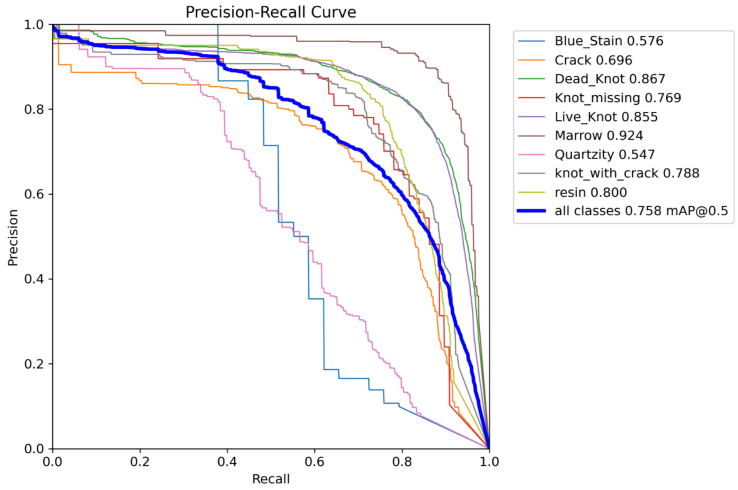


Figure 3.13: Precision-recall curve in a multi-label object detection setting. The nature of the curve undermines the inverse relationship between precision and recall. The dark blue curve denotes the mean average precision over all classes at an IoU threshold of 0.5.

3.5.3 Runtime Evaluation Metrics

Runtime performance metrics measure parameters such as GPU memory usage, GPU utilization, and training time. These metrics often depend on the hardware used and therefore do not give an absolute indication. However, they can give clues about the performance requirements during training and inference. In addition to hardware, the choice of input image resolution and architecture influence the memory and time requirements. Deep accurate backbone networks often lead to slower inference times (Lin, Goyal, Girshick, He & Dollar 2017). While the metrics do not say anything about the quality of the models themselves, the trade-off between speed and accuracy may be important for users where the availability of powerful hardware is relevant (Huang et al. 2017).

Frames per Second

The speed of object detectors is measured in frames per seconds (FPS), i.e. the number of frames or images that can be processed by an object detector in one second. Here, higher FPS values correspond to a faster processing speed. (Bochkovskiy et al. 2020) define object detectors that achieve speeds of 30 FPS or more as *real-time detectors*. FPS can be influenced by various factors, including the complexity of the object detection model, the computational resources available, and the resolution of the input frames. Achieving high FPS requires optimizing both the architecture and implementation of the object detector. Techniques like model compression, parallel processing, and hardware acceleration can be employed to improve FPS without sacrificing accuracy significantly.

In this work, the FPS for the object detection algorithms are determined when evaluating the model with the test dataset. For this, the batch size is set to one and the time to process one image is determined. For a better adaptation to the use case in a production context, a plot of the detection is simulated after each processing. The total processing time consists of the preprocessing time (PRE), inference time (INF), and postprocessing time (POST), each in ms. Hence, the FPS value is computed as follows:

$$\text{FPS} = \frac{1000}{PRE + INF + POST}$$

Floating-point operations (FLOPs) are a measure for the complexity of the model that do not depend on the underlying hardware (Huang et al. 2017).

Chapter 4

Related Work

In this chapter, the landscape of existing research is examined. For this, a review of relevant studies that have explored various approaches with different technologies and methodologies to address wood defect detection is provided. This chapter seeks to identify gaps in research as well as trends which lays the foundations for this work.

4.1 Wood Defect Detection

In 2003, Silvén et al. (2003) developed a non-supervised clustering method for detecting defects in lumber boards. For this, a self-organizing map (SOM) algorithm is used. The approach involves two-stages. First, a human expert manually separates regions suspected of having defects from intact regions on the surface by setting boundary lines on the computer. Second, the feature values of the defective regions are merged into a vector and then classified via a detection SOM. The authors emphasize that the quality of the results is strongly influenced by human interaction. The choice of boundary regions significantly influences the results in terms of the amount of non-detected defects (error escapes). The highest difference in terms of error escape were 38.9 % vs. 13.3 %. Moreover, as a human operator decides to classify a region as defective or not in the first place, this methodology is still prone to human error (Silvén et al. 2003).

Kline et al. (2003) utilized a multiple sensor lumber scanning system to automate the grading of hardwood lumber. This continues the research conducted by Conners et al. (1997). The prototype system that was used, incorporates a color line scan camera system, a laser-based ranging system, and an X-ray scanner as part of its setup. During the scanning process, the system generates merged images from objects moving on a conveyor belt. The system outputs information about the type of each identified defect, as well as its location and extend. This so called

defect map is generated by software that initially preprocesses the images and then segments the image into regions of interest, followed by the use of a rule-based fuzzy logic with custom membership functions for the classification task. The system was trained on a data set with 300 images, and at least 40 images for each of the ten different defect types. On unseen test data, the system and lines graders achieved a classification accuracy of 63 % and 48 %, respectively. This means, the automated system performed 31 % better than the manual graders in terms of accuracy. However, in the study, small defects posed challenges in terms of online detection, and discolorations of the material led to false positives (Kline et al. 2003).

A different approach with the assistance of computational intelligence techniques was proposed by Estévez et al. (2003) in 2003. The approach utilized a genetic algorithm in combination with a neural network for the classification of defects on wooden boards. For the study, 2958 samples of wood surfaces were used. The database was created using video camera color recordings of material surfaces, which were manually labeled based on the largest defect, allowing each sample to be assigned to one of the ten specific defect classes or the negative class without any defects. The samples were captured multiple times in various orientations. The samples were divided into defective and intact regions using a histogram-based segmentation module, where defective regions were represented as objects using an overlaid window. The features were then extracted by a feature extraction module and selected by a genetic algorithm. Subsequently, the data were classified into one of the categories using a multilayer perceptron (MLP) with one hidden layer. The study yielded classification accuracies of 74.5 % for all defect classes and 87.8 % for a reduced problem with only 7 classes (Estévez et al. 2003).

Computational intelligence methods for segmentation were further utilized by (Estevez et al. 2005) and (Ruz et al. 2009). The studies employed fuzzy min-max neural networks for generating minimal bounding boxes around objects as a segmentation algorithm. These bounding boxes, here called hyperboxes, are generated from automatically detected seed pixels. Ruz et al. (2009) increased the number of features compared to Estévez et al. (2003) and performed the classification with SVM in addition to the originally proposed MLP. The use of additional features improved the classification results. In the study, the pairwise classification via SVM demonstrated better results than the MLP classification and lead to an average classification rate of 91.39 % (Ruz et al. 2009).

Chacon & Alonso (2006) presented another method for defect detection using fuzzy logic. In their study conducted in 2006, four types of knot defects are classified. Gabor filters are used for feature extraction. The study compares the performance of unsupervised and supervised neural networks, specifically using a self-organized neural network (SONN), a SONN with fuzzy parameters, and a feedforward perceptron neural network (FFPNN). The authors report a test accuracy of 85.28 %, 88.23 %, and 91.17 % for SONN, fuzzy SONN, and FFPNN, respectively

(Chacon & Alonso 2006). However, the results should be interpreted with caution. Only 68 and 72 samples were used for training and testing, respectively. Additionally, for all models, the reported training accuracies surpass the reported test accuracies. This indicates overfitting, which implies that the model has memorized the training data but struggles to generalize well to unseen data.

In 2009, Gu et al. (2009) introduced the use of support vector machines (SVM) for classifying different types of knots on lumber boards. The algorithm used represents a four-stage tree structure, where an SVM classifier at each branch decides whether the defect belongs to one of the four classes based on a specific feature of the feature vector. The classifier is trained with 800 images and then tested with 400 unknown instances. The images of lumber boards are initially reduced to rectangular patches, containing only the defect and small parts of the background. The method used achieves an average classification rate of 96.5 %, and a false positive rate of 2.25 % (Gu et al. 2009).

A faster method for defect detection on wood veneer surfaces was presented by Shi et al. in 2020 by using a multi-channel mask region-based convolutional neural network (Mask R-CNN) (He et al. 2017) with the addition of a glance network. The faster detection is achieved by preselecting the images with defective surfaces by the glance network. This achieved that only relevant data need to pass through the actual detection algorithm. Object detection on the preselected data was then performed by a mask R-CNN with ResNet backbone. With this method, an accuracy of more than 98 %, and a mAP value of about 95 % could be achieved at a processing speed of 2.5 s per 100 frames. However, the study was limited to the detection of only three defect types (Shi et al. 2020).

Mask R-CNN was also proposed by Li et al. (2021) in 2021 to detect three types of defects on wood surfaces. The image data were captured via a common DSLR camera. A cycle generative adversarial network (cycle GAN) (Zhu et al. 2017) was used to enhance the initial data set of 1242 images with a total of 1691 present defects. The authors used this approach to counter the problem of imbalanced defect data as knots make up roughly 60 % of the defects in the images. Using this approach, 430 synthetic images were added to the data set resulting in 1673 images, and 2414 defects present. In addition, data augmentation techniques were used. The ResNet backbone of the Mask R-CNN was modified by introducing layered connections. Additionally, the initial 3x3 convolution was replaced by so-called deformable convolution (Zhu et al. 2017, 2019) to account for geometric variations of the input defects. Deformable convolutions allow the standard convolution to freely deviate from its grid-form in 2D space. Supplementary convolutional layers learn the offsets from the feature maps that come before them. The proposed model achieved mAP values of about 84 % and 83 % for bounding box and mask information, respectively. The model used had a processing speed of 14.83 FPS (Li et al. 2021).

Hwang et al. (2021) propose a traditional artifical neural network (ANN) architecture with external feature extraction to detect different types of knot defects. The authors use texture descriptors and local feature descriptors for feature extraction from a data set of 937 images of knot defects. Grey-level co-occurence matrix (GLCM) and local binary pattern (LBP) approaches are used for the texture descriptors approach. Histogram of oriented gradients (HOG), scale-invariant feature transform (SIFT), and dense scale-invariant feature transform (DSIFT) are used as local feature descriptors. The ANN is build as a feed-forward network with multiple layers. For comparison purposes, a k nearest neighbor (k-NN) and support vector machine (SVM) are trained on the same input features. The texture descriptors yield better results than the local feature descriptors. The ANN model achieves a F1-score of 0.793 and 0.613 for GLCM and LBP, respectively, and 0.776, 0.711 and 0.718 for HOG, SIFT, and DSIFT, respectively (Hwang et al. 2021).

4.2 Transfer Learning in Wood Defect Detection

In 2015, Norlander et al. (2015) proposed a convolutional neural network architecture similar to that of AlexNet to detect wooden knots on wood surfaces. For defect localization, a siding window approach was used. The algorithm was pre-trained on ImageNet and then learned using training data from the target domain. For this purpose, deep layers of the pre-trained network were replaced by new, trainable layers, while the shallower layers were frozen. Subsequently, fine-tuning of all layers was performed with a lower learning rate. The network pre-trained on ImageNet achieved a higher accuracy than the variant without transfer learning. However, the method used only distinguishes between knot (positive) and wood (negative) classes, other defects are not detected (Norlander et al. 2015).

In 2019, Urbonas et al. (2019) proposed Faster R-CNN for the detection and localization of defects on wood veneer surfaces. The method was employed to identify four types of surface defects using common CNN architectures with transfer learning as backbones. A data set of 353 images was acquired by scanning the veneers with a line scan camera on a conveyor belt system. The data set was artificially augmented to 3530 images for training and testing. For recall, precision, and F1 scores, values of up to approximately 0.8 were obtained in the best case. The averaged accuracy reached around 80 % in the best case. The processing speed in this case was around 40 ms (Urbonas et al. 2019).

Another method for the detection of defects on veneer wood surfaces was proposed by Hu et al. (2020) in 2020 via Mask R-CNN and transfer learning. The study aimed at the detection and localization of three different defect types. A total of 1050 image data were acquired via industrial

cameras and synthetically augmented by 100 instances of each defect class using a GAN. Each instance showed a defect in close-up, negative examples without defects were not used in the analysis. The model pre-trained on COCO achieved an increase in accuracy of about 4 % compared to the from scratch model. The study showed a maximum achieved accuracy of 94.7 % and a maximum achieved mAP value of 98.4 %. The limitation of the study to close-up images of defects, however, makes the transferability to real scenarios questionable (Hu et al. 2020).

An approach for classifying and locating defects with a fully convolutional neural network was proposed by He et al. (2019) in 2019. The authors introduced a so-called mixed, fully convolutional network (Mix-FCN) for semantic segmentation that consists on components of the VGG and InceptionNet architectures. The weights of the network were transferred from one VGG network pre-trained on another data set from the same domain. The model was then trained on a new dataset with six different defect categories. All data used were acquired using a line scan camera, using two different wood species in equal proportions. The data was then augmented using different augmentation techniques. The proposed method achieved scores of 99.14 %, 91.31 % in terms of overall classification accuracy and pixel accuracy, respectively. The detection speed was reported to be 368 ms per batch of 50 images (He et al. 2019).

Ding et al. (2020) used an ImageNet pre-trained modified SSD one-stage detector with DenseNet backbone for object detection of three different defect types on solid wood panels. A line scan camera was used for data acquisition. The acquired images were then cropped into 500 square sections of defects. Augmentation methods were applied to enhance the training data. The results of the study are reported to be about 96 % and 56 ms for mAP and processing time, respectively (Ding et al. 2020).

Another one-stage object detection approach with a modified YOLOv3 (Redmon & Farhadi 2018) algorithm was proposed in 2021 by Tu et al. (2021) to detect four types of defects in sawn lumber from two different wood species. The image data were acquired by using an area array camera system scanning moving lumber on a conveyor belt. The data set used consists of 5840 and 1108 images of rubber lumber and pine lumber, respectively. The data sets of the two species had different types of defects. The pine lumber data set included only two types of defects, while the rubber lumber data set contained two additional types of defects. The highest measured mAP value of 86 % was achieved at a speed of approximately 38 FPS.

A solution for multi-class defect classification using pre-trained CNN was presented by Gao, Qi, Mu & Chen (2021) in 2021. The authors use a ResNet-34 pre-trained on ImageNet to identify seven different knot defects from a data set of 450 defect close-up images, which was artificially enlarged using augmentation methods. The results of the pre-trained and modified architecture were compared with the results of the base architecture without transfer learning. Compared to the training from scratch, the accuracy of the ResNet-34 could be improved by about 1.5 % using

transfer learning. The convergence speed also shows improvements with the use of transfer learning (Gao, Qi, Mu & Chen 2021). Gao, Song, Wang, Liu, Mandelis & Qi (2021) could further improve the results on the same dataset by using a modified ResNet-18 CNN with attention mechanism in combination with transfer learning. In direct comparison to the non-pre-trained model, accuracy improved by over 2 % when using transfer learning. The convergence speed also improved (Gao, Song, Wang, Liu, Mandelis & Qi 2021). Gao et al. (2022) proposed a slightly modified version of the algorithm using more attention modules. This version demonstrated comparable results when using transfer learning. However, the pre-trained model only achieves an accuracy improvement of about 0.3 % compared to the model without transfer learning (Gao et al. 2022).

Chapter 5

Methodology

5.1 Data

Data is one of the most critical elements in computer vision. This section addresses the data used for the study which serves as a fundamental basis for evaluating the research questions. First, the origin and acquisition of the data are described. Second, an overview about the composition of the data and meta-information are given in more detail. Finally, the preprocessing and transformation steps performed on the data are explained.

5.1.1 Data Acquisition from Database

Data from Kodytek et al. (2022) are used for the analysis and experiments of this study. The images were taken on a production line in a sawmill under real production conditions. The images demonstrate sections of lumber surfaces in plan view. The authors of the paper make the data publicly available for download at Kodytek Pavel et al. (2021). The available dataset consists of 20.276 images in .bmp format, the respective samples associated labels and bounding box coordinates in .txt format, the respective semantic map information in .txt format, and meta-information about semantic maps. The labeling of the samples was performed by the authors of the paper using a self-developed tool (Kodytek & Bodzas 2021) which is also provided for download. The samples have a resolution of 2800 x 1024 pixels. Figure 5.1 shows some examples of the image data from the data set.

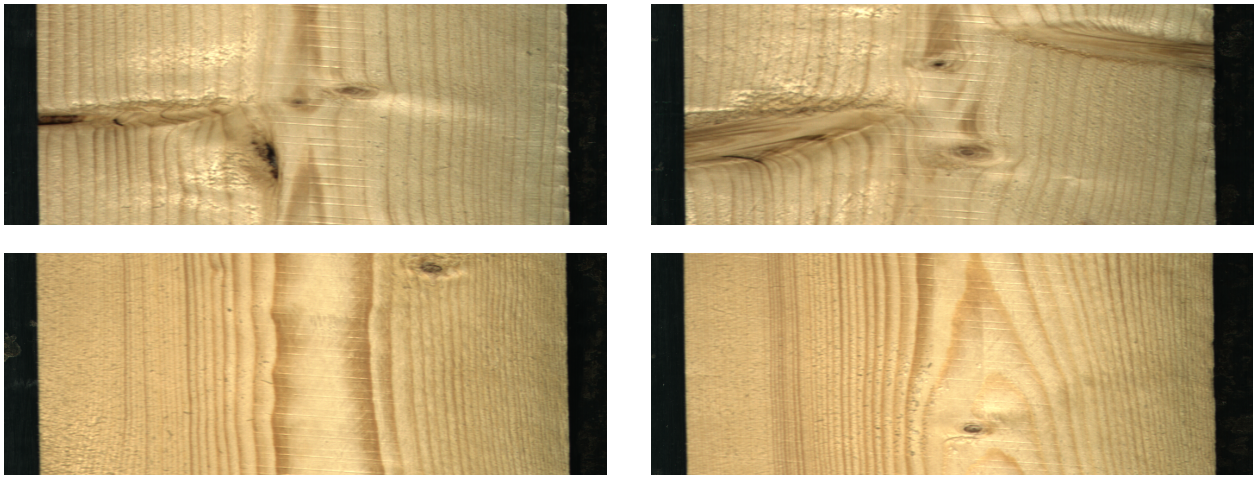


Figure 5.1: Examples of the image data downloaded from (Kodytek Pavel et al. 2021).

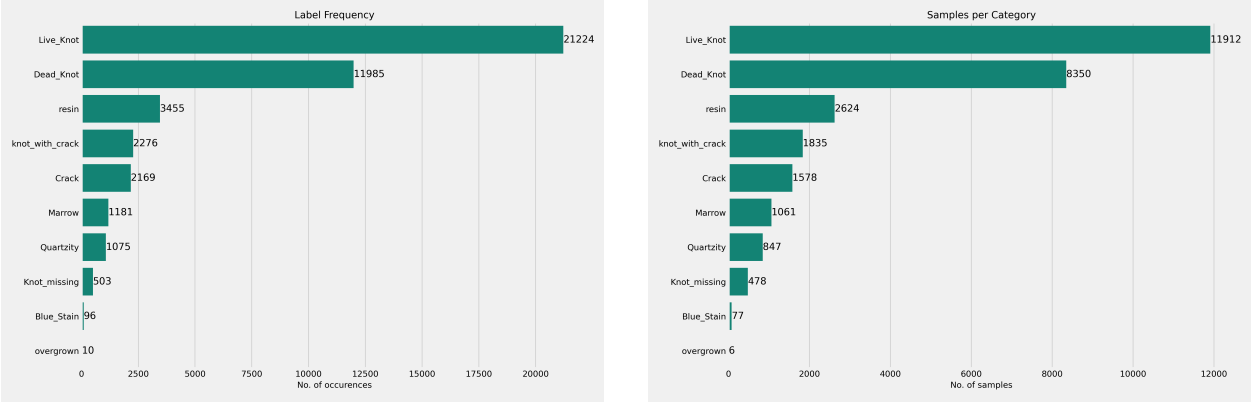
5.1.2 Data Analysis

Each sample contains none, one, or multiple defects. According to (Kodytek et al. 2022), 18.284 samples contain defects, and 1992 samples are without any defects. A total of ten different defects are covered by the image data. The defect types present are: Blue stain, crack, dead knot (decayed knot), knot hole, live knot, pith, mineral streak, knot with crack, overgrown, and resin. The associated labels used are: Blue_Stain, Crack, Dead_Knot, Knot_missing, Live_Knot, Marrow, Quartzity, knot_with_crack, overgrown, resin. To avoid confusion, the class names chosen by the authors will be adhered to in the further development of this thesis.

It should be noted that the data set is not balanced. Some defect classes such as live knot are highly prevalent in the data set, whereas defects such as overgrown, and blue stain are extremely rare. Table 5.1 shows the total number of the respective defect class, the total number of the respective defect class in percent, the number of samples on which the defect class is represented at least once, and the percentage ratio to the total number of samples.

5.1.3 Data Preprocessing

Data preprocessing techniques are applied to facilitate the training with the models proposed in this thesis. In addition, the data is processed to lower the requirements for computational resources and the training time of the models. Different preprocessing steps are applied to the data set described in section 5.1. The different steps for transformation, cleaning, and organization that are performed on the data are described in the following.



(a) (b)

Figure 5.2: Bar plots representing the distribution of defect labels. Figure (a) displays the label frequency by defect label in descending order. Figure (b) presents the number of samples by defect label in descending order. This showcases the imbalance of the data set in terms of defect labels. The background class is excluded.

Table 5.1: Occurence of defects.

Defect class	No. of label occurences	Percentage occurence	No. of samples with defect	Percentage of total samples
Live knot	21224	48.26 %	11912	58.75 %
Dead knot	11985	27.25 %	8350	41.18 %
Resin	3455	7.86 %	2624	12.94 %
Knot with crack	2276	5.18 %	1835	9.05 %
Crack	2169	4.93 %	1578	7.78 %
Marrow	1181	2.69 %	1061	5.23 %
Quartzity	1075	2.44 %	847	4.18 %
Knot missing	503	1.14 %	478	2.36 %
Blue stain	96	0.22 %	77	0.38 %
Overgrown	10	0.02 %	6	0.03 %
Sum	43974	100 %	28768	

Resizing

The data of (Kodytek et al. 2022) is available in a resolution of 2800 × 1024 after download. This resolution is not suitable for most algorithms, since large amounts of memory and computation are required for processing. To counteract these problems, the image files are scaled down to 700 × 256. The aspect ratio, i.e. the ratio of width and height of the images, is not changed. This is to avoid distortions of and to maintain the visual representation of the defects and the wood surface. It should be noted that the object detectors of the YOLO family only accept resolutions that are a multiple of the max stride of 32. The data is therefore scaled to a resolution of 704 during training and validation.

Label Transformation

The coordinates of the bounding boxes are available in the Albumentations (Buslaev et al. 2018) format after the download. In the Albumentations bounding box format, each bounding box is represented as a tuple of four values [class_name, x_min, y_min, x_max, y_max]. These coordinates are normalized, i.e. they are scaled by the width and height of the image and range from 0 to 1. For the use of the data with the models of the Yolo family, they are converted into the format [class_id, x_center, y_center, box_width, box_height]. Again, the values are normalized based on the dimensions of the image. Futhermore, the class names in the first place were encoded with unique numbers from 0 to 9.

Cleaning

The present data have some defective bounding boxes with minimal area that do not enclose any defect. These are removed from the label files during the downscaling step.

Offline Data Augmentation

To evaluate the impact of offline augmentation, data augmentation techniques are used to enhance the given data set. In theory, this should improve the generalization capability of the trained models and increase their robustness and performance (see section 3.1.1). In contrast to online data augmentation, offline data augmentation increases the size of the data set. For this study, offline augmentation is applied to the training and validation data set. The test set is not augmented. The following five augmentation techniques are applied independently of each other to the original data:

- Vertical flipping: All images are vertically flipped. In this geometric transformation, the image is mirrored on the horizontal axis. As a result, the positions of objects, patterns, and features within the image are reversed vertically. Vertical flipping is used to make to model more robust towards changes in the position and orientation of objects. Figure 5.3(b) illustrates a vertically flip of the original image 5.3(a).

- Hue shift: A random hue shift between -25 and +25 degrees along the color wheel is applied to the images. This changes the color representation of the images without affecting brightness or saturation parameters. Hue shift is applied to make the model more invariant to changes in color that can occur due to lighting conditions or other real-world challenges. Figure 5.3(c) demonstrates a hue shift of -20° applied to the original data.

- Saturation: Random adjustments in the color vibrancy are applied to the images. The adjustments range between -75 % and 75 % with respect to the original image. This technique is used to improve the model's ability to generalize to new environmental conditions. Figure 5.3(d) shows an adjustment of saturation by -75 %.

- Gaussian noise: Random noise is added to the image pixel values. The noise values are drawn from a Gaussian distribution with zero mean. The standard deviation is set to a value of 10. The standard deviation controls the intensity of the noise. Noise on images can occur due to factors such as lighting, or sensor defects. Training the model on these conditions can make the model more robust to variations in the inputs. Figure 5.3 (e) represents the modification with Gaussian noise.

- Salt and pepper noise: This augmentation technique randomly adds white and black pixels to the image. This is done by randomly selecting pixels in the image and changing their color to white or black based on a predefined probability. The probability value is set to a value of 1 %. This controls the intensity of the noise. The salt and pepper augmentation technique is used to make the model more robust to imperfections occurring in the input data. Figure 5.3(f) shows an example of added salt and pepper noise.

Organization

The data are randomly divided into training, validation and test set in advance. A percentage distribution of 60 %-20 %-20 % is used for the training, validation, and test set, respectively. The data is stored in a specific folder structure for training the YOLO algorithms. Each of the training, validation and test directories contains a subdirectory for images and labels. In the case of the training and validation directories, there are further subdirectories for images and labels of the originals, and each of the respective augmentation techniques applied.

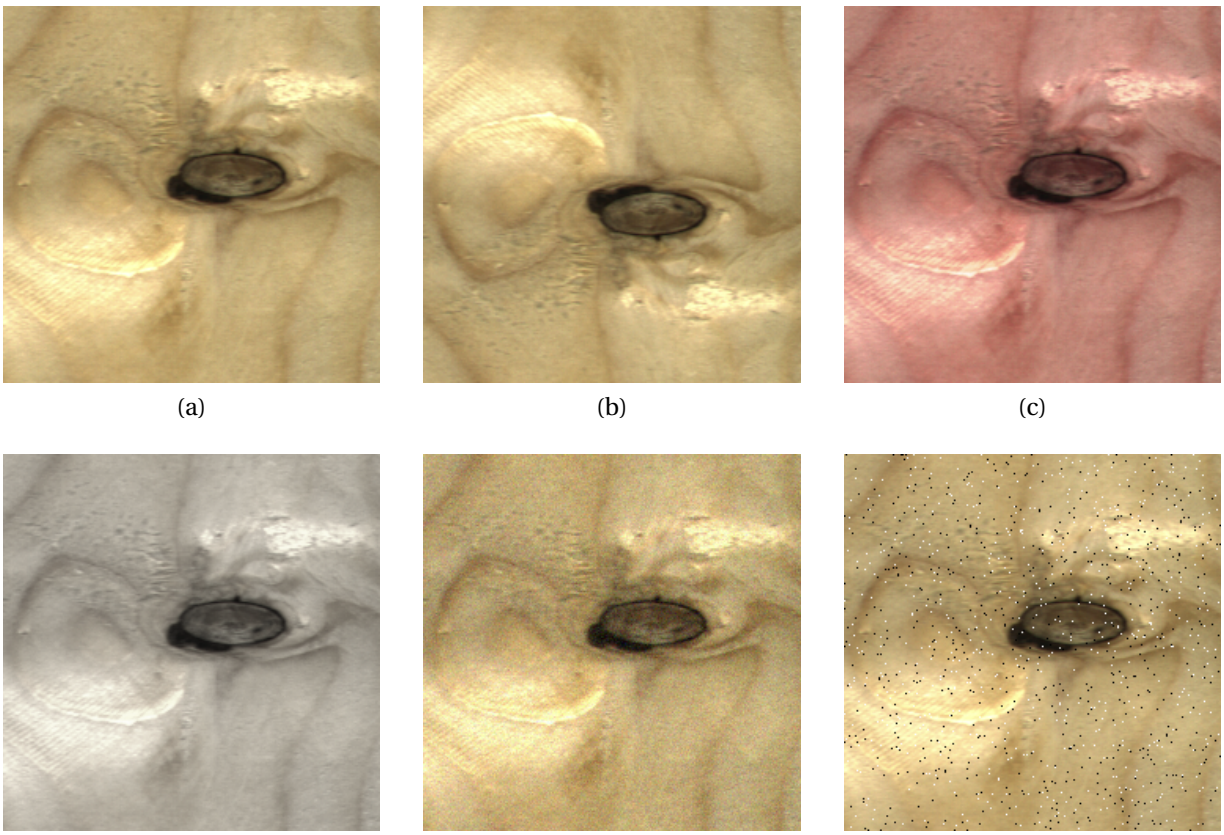


Figure 5.3: Examples of the different data augmentation techniques applied. Cropped parts of the images are used for representation purposes. (a) Original, (b) vertically flipped, (c) hue shift, (d) saturation, (e) Gaussian noise, and (f) salt and pepper noise.

Table 5.2: Composition of raw data set.

Class label	Train	Val	Test
Blue stain	46	29	21
Crack	1311	451	390
Dead knot	7095	2435	2350
Knot missing	326	87	78
Live knot	12555	4178	4300
Marrow	687	247	239
Quartzity	606	198	207
Knot with crack	1322	427	451
Overgrown	5	0	0
Resin	2081	678	682
Background	1227	382	384

Table 5.3: Composition of offline augmented data set.

Class label	Train	Val	Test
Blue stain	276	174	21
Crack	7866	2706	390
Dead knot	42570	14610	2350
Knot missing	1956	522	78
Live knot	75330	25068	4300
Marrow	4122	1482	239
Quartzity	3636	1188	207
Knot with crack	7932	2562	451
Overgrown	30	0	0
Resin	12486	4068	682
Background	7362	2292	384

5.2 Implementation Details

Details of the hardware and software and the frameworks used are provided in this section. The hardware and software provide the basis for processing the data and training the computer vision models presented in Section 3.3. The information about hardware and software used is important for the subsequent interpretation of the results and also for their reproducibility. Furthermore, the choice of hardware and framework are relevant with respect to the potential feasibility of the application within a real production setup.

5.2.1 Hardware Used

For storing and pre-processing the data as well as training and evaluating the models, resources of the Idun cluster, a professionally managed computing platform of NTNU, are used. As conventional mobile hardware is not suitable for training large computer vision models, the training is performed on one or more GPUs of the cluster depending on the task. In the following, details about the hardware used by type of computer vision task are provided.

Six object detection models are compared in this evaluation. To ensure the comparability of the benchmarks, all object detection models, i.e. all scaled versions of the YOLOv7 and YOLOv8 family are each trained and evaluated on a single NVIDIA A100-PCIE GPU with 40 GB VRAM. CUDA version 11.7 is used. 60 GB of RAM are allocated for each run. As CPU, an Intel Xeon Gold 6248R with 48 cores is used.

The three CNN models for image classification are each trained on two GPUs of type NVIDIA A100-PCIE with 80 GB VRAM and CUDA version 11.7. As CPU, Intel Xeon Gold 6248R with 48

cores or AMD EPYC 75F3 with 64 cores are used, depending on the assigned node. As for the object detection models, 60 GB of RAM are allocated for each run.

5.2.2 Software Used and Modifications

All models are trained and evaluated on a Red Hat Enterprises Linux distribution in version 4.18.0-425.3.1.el8.x86_64. Resource management and job scheduling are performed via Slurm Workload Manager. Specific Anaconda virtual environments are created to avoid compatibility conflicts between dependencies. Python version 3.9.16 is used for all of the virtual environments.

The YOLOv8 models are based on ultralytics version 8.0.153. Both frameworks use torch version 2.0.1. The frameworks were not modified for this evaluation.

For the CNN implementations, the TensorFlow framework (Abadi et al. 2015) version 2.12.0 and the TensorFlow Keras (Chollet et al. 2015) API are used. EfficientNetV2B0 and ResNet-50V2 are imported directly from Keras Applications. Since the API does not have a ResNet-18 implementation, this is imported from the Classification models Zoo library (Iakubovskii 2018). In all cases, the original output layers of the models are substituted with a custom head (see figure 5.4 for reference). The custom head consists of a 2D average pooling layer, followed by a flattening operation. Following this, a fully connected layer comprising 512 neurons, L2 regularization and activated by ReLU is incorporated. Subsequently, a dropout layer with rate 0.5 is implemented. The final output fully connected layer is using a Sigmoid activation function. The width of the output layer is determined by the number of classes.

5.2.3 Choice of Hyperparameters

Due to the limited time for this thesis and the computational complexity, the choice of hyperparameters is determined by trial and error. For this purpose, different parameter settings are tested and evaluated based on training and test performance. For the sake of simplicity, only the results of the best models are presented in this analysis and no further comparison between different hyperparameter setting is made.

Hyperparameters for Object Detection Models

All object detection models are trained using the same set of hyperparameters, without distinctions between the different analyses related to research questions. The hyperparameter settings

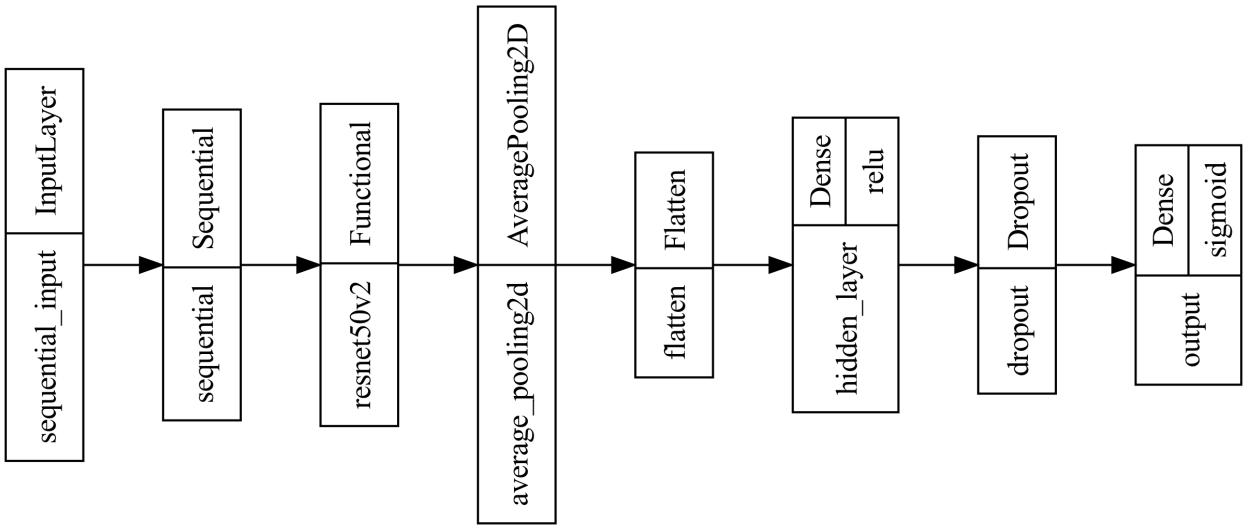


Figure 5.4: High-level visual representation of the CNN model architecture with the incorporated custom head. The sequential layer after the input layer denotes the data augmentation layer. This is followed by the main architecture, in this case ResNet-50V2. The final five layers denote the custom head attached to the model.

for the models in the YOLOv8 family, as well as the models in the YOLOv7 family, remain consistent across the different scales of the models used. For all models, a batch size of 32 is used during training. Each model is trained for 200 epochs, or until early stopping (ES) is triggered in the case of YOLOv8. YOLOv7 does not incorporate an early stopping function. For training on the use case data, the maximum number of epochs is increased to 300. The standard hyperparameters of the models are otherwise employed (see the GitHub repositories for YOLOv7 (Wong 2022) and YOLOv8 (Jocher et al. 2023) for reference). Stochastic Gradient Descent (SGD) serves as the optimizer. The initial learning rate (lr0) is set at 0.01, and the final learning rate (lrf) is determined by lr0 × lrf. For online data augmentation, the standard data augmentation settings of the algorithms are utilized. An overview of the key hyperparameters is provided in Table 5.4.

Table 5.4: Hyperparameter settings for YOLOv8 and YOLOv7 models. The maximum number of epochs is different for the original data set and the use case data set.

Model	Batch S.	Opt.	#Epochs$_{OD/UCD}$	lr0	lrf	Momentum	Weight Decay
YOLOv8	32	SGD	200(ES) / 300(ES)	0.01	0.01	0.937	0.0005
YOLOv7	32	SGD	200 / 300	0.01	0.1	0.937	0.0005

Hyperparameters for Image Classification Models

All CNN models in this thesis are trained with a batch size of 32 per GPU. The Adam optimizer is used, with the initial learning rate being reduced over the course of training using cosine decay schedulers. The number of scheduler steps is determined by the product of the batch size and the size of the training data set. All models are trained for a maximum of 200 epochs on the original data, and 500 epochs on the use case data. In addition to that, early stopping is implemented in all models, starting at epoch 20. If no improvement in the validation loss occurs over the subsequent 50 epochs, training is stopped, and the weights from the best epoch are restored. The binary cross-entropy loss function is utilized. Additionally, online data augmentation is performed by applying random geometric variations to the inputs: random horizontal and vertical flip, random rotations ranging from -18° to +18°, and random zooming ranging from -10 % to +10 %. No other augmentation techniques are employed. An overview of the hyperparameters used for the image classification models can be found in Table 5.5.

Table 5.5: Hyperparameter settings for ResNet and EfficientNet models. The maximum number of epochs is different for the original data set and the use case data set. lr_{OD} and lr_{UCD} refer to the learning rates used for training on the original data set and the use case data set, respectively.

Model	Batch S.	Opt.	$\#\text{Epochs}_{OD/UCD}$	$\text{lr0}_{OD/UCD}$	Dropout	Weight Decay
ResNet-18	32	Adam	200(ES) / 500(ES)	5E-4 / 1E-5	0.5	0.01
ResNet-50V2	32	Adam	200(ES) / 500(ES)	5E-4 / 5E-6	0.5	0.01
EfficientNetV2B0	32	Adam	200(ES) / 500(ES)	1E-5 / 1E-4	0.5	0.01
EfficientNetV2S	32	Adam	200(ES) / 500(ES)	1E-5 / 1E-4	0.5	0.01

Chapter 6

Use Case Application

6.1 Use Case Introduction

The use case of this thesis aims to prepare the implementation of a system for surface defect detection in a production for window frames. In the production, windows are manufactured whose frames are made of wood. The defect detection system is planned to be installed in perspective behind the first sawing machine of the production line. There, delivered wooden parts that have already been cut into shape are sawn into different lengths according to the order. Four of these cut pieces are needed for each window, one for each side. The defect detection system will inspect the sawn parts for defects after the initial sawing process. The defect inspection system is to replace the manual inspection at this station in the production. The use case is intended to investigate whether the algorithms and training methods investigated in this thesis would be suitable for such a task.

Due to delays in the delivery of the necessary hardware as well as scheduling conflicts with the partner company, the experimental setup cannot be implemented and tested directly in the production there. Instead, a prototype system is developed in the learning factory of the university's workshop laboratory. This setup is thoroughly evaluated and prepared for possible implementation in practice. In this chapter of this thesis, a comprehensive overview of the practical implementation conducted in the learning factory environment is provided.

6.2 Experimental Setup

This section introduces the laboratory environment and the setup of the line scan camera system for data acquisition. For this, details about the equipment used and the necessary software

are provided. Important settings and parameters that are important for reproducing the results are also determined and recorded. In addition to the general conditions, the procedure for image data acquisition is explained. This is followed by information on the labeling process of the acquired data and descriptions of the transformation and pre-processing steps.

6.2.1 Laboratory Environment

The experimental setup takes place in the Cyber-Physical Learning Factory at the NTNU campus in Gjøvik. Experimentation in the learning factory will be conducted to simulate the implementation of the system in an actual production environment. At the time of the study, such an implementation was not possible. The learning factory is part of the national research infrastructure MANULAB and is used for research purposes in the field of production engineering. The learning factory represents a full-size production line and has various conveyor systems, industrial robot cells, sensor technology, HMI, as well as the associated software. Additional equipment such as cameras and exposure systems can be installed on the production line with the help of a modular system. The conveyor belt runs at a fixed speed of 110 mm per second.

6.2.2 Equipment and Software Used

This section provides information about the hardware and software used within the experimental camera setup for data acquisition.

Camera Hardware

For the practical test, a color line scan camera of type FS-C2KU7DGES-C from the manufacturer Omron Sentech is used. The camera uses the GigE interface standard and can be powered by Power over Ethernet (PoE). The camera's sensor has a resolution of [2048 × 2] (2k) with a pixel size of 7 µm. The scan frequency of the camera is 26 kHz. For better light sensitivity, the camera has a dual-line system, i.e. the sensor has two rows of pixels instead of just one. This allows the signal to be vertically combined without reducing the effective resolution. For the experiments, a lens of type Omron 3Z4S-LE VS-2514H1 is used which is mounted to the camera via C-mount. The lens has a focal length of 25 mm. The minimum distance is specified as 30 mm. The Field of View (FOV) is 25.1° × 25.1° ($H \times V$).

Camera Software

As software, the Sentech SDK Package (v1.2.1) with version macOS1x_Intel_v1.2.1 is used initially. The package is installed on a Dual-Core Intel MacBook Pro 2016 running macOS Monterey (v12.6.8). It should be noted that the computer used only allows a packet size of 1500 Maximum Transition Units (MTU). According to the manufacturer, so-called jumbo frames, i.e. packet sizes of 9000, are recommended for efficient transmission.

6.2.3 First Test Setup

The initial test setup is shown in Figure 6.1. In the initial test setup, cut-to-size wooden parts are attached to platforms designed for transport on the conveyor belts. The line scan camera is positioned at a distance of 300 mm from the workpiece surface using a modular aluminum construction. The distance of the light source to the workpiece surface is 140 mm. The light source illuminates the workpiece surface from an angle of 45° relative to the horizontal.

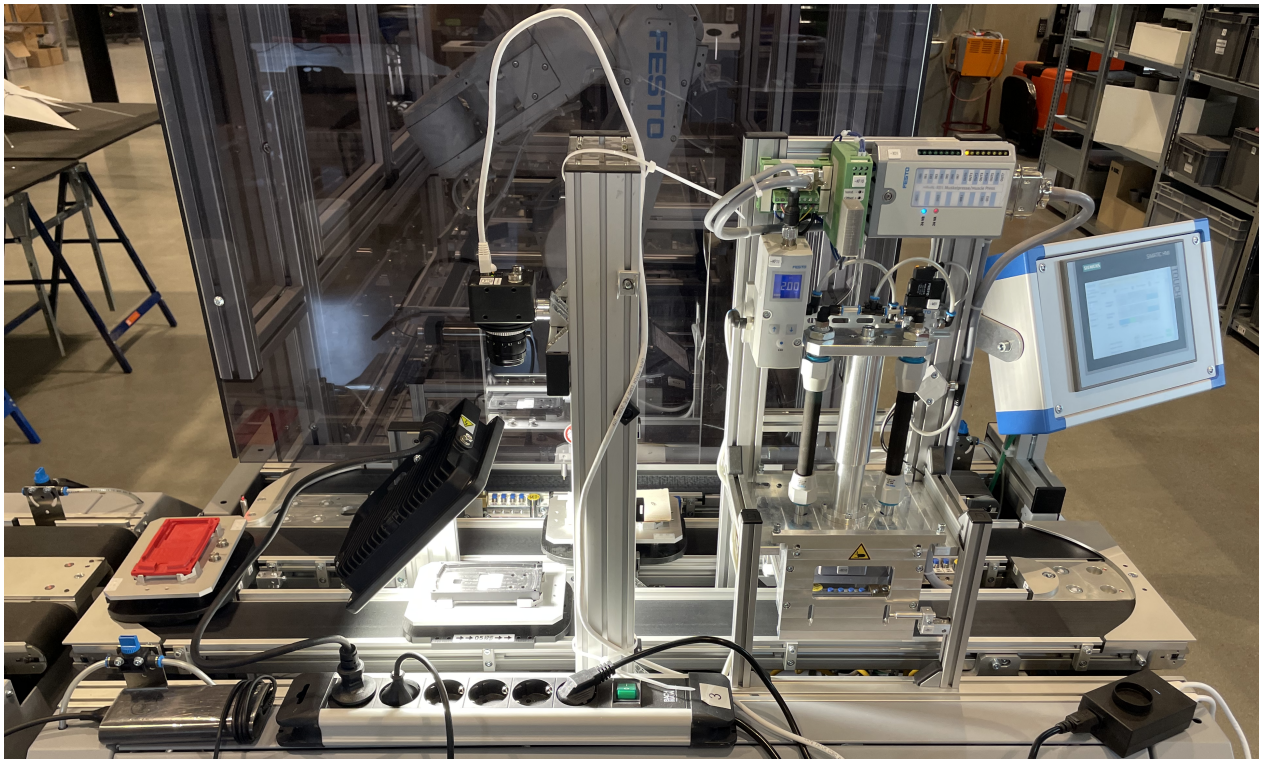


Figure 6.1: Initial test setup with activated illumination. The samples are placed on the moving platforms.

In the initial test setup, the light unit is missing due to supply shortages. As a substitute, an LED worklight is used and installed at an angle of 45° relative to the test object. However, in this test

setup, artifacts appear in the camera image during test runs, which are presumably due to the 50 Hz flickering of the light source. The artifacts appear in the form of black horizontal bars that can be seen at regular intervals on the camera's output stream. (Figure 6.2. In addition, it appears that the luminosity of the spotlight is not sufficient for a result sufficient for the purposes of this thesis.

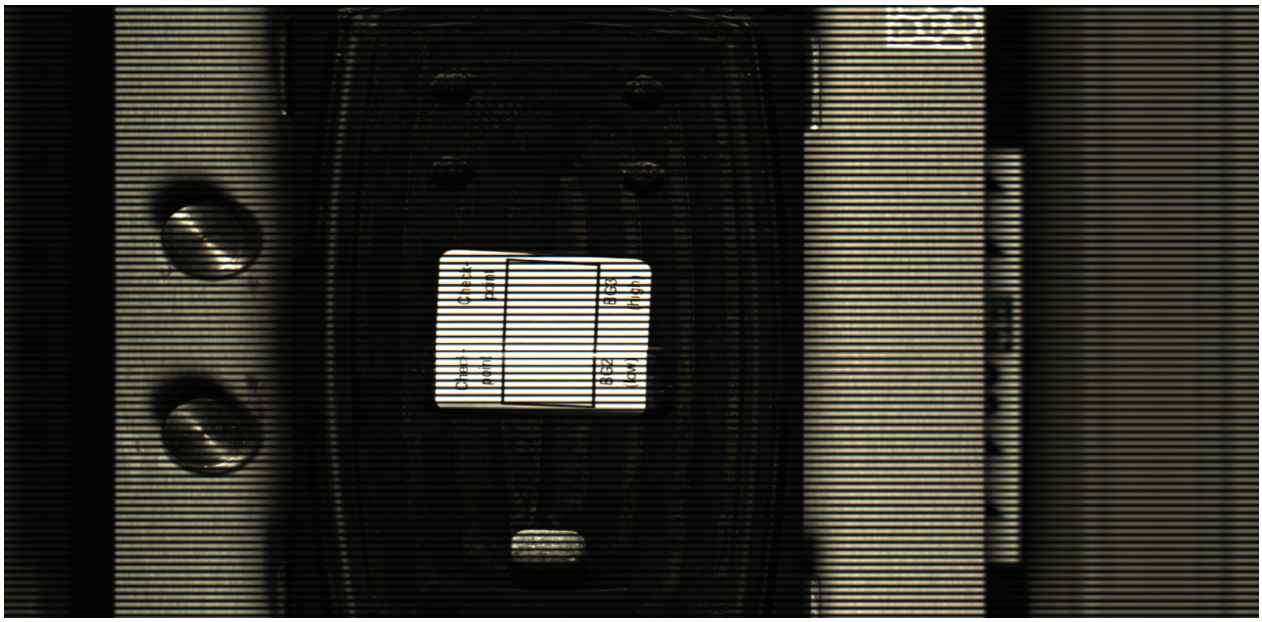


Figure 6.2: Test results with inital setup. The black horizontal stripes originate from the 50 Hz flickering of the LED worklight.

6.2.4 Final Test Setup

For the final setup, changes are made to fix the previously mentioned problems. The previously used worklight is replaced by an industrial white LED bar of type M-EBAR-125-WHI-UN which is connected to a power supply of the production line. The bar light has an active area of 125 mm. The lighting unit is installed above the conveyor belt by means of a self-made structure made of aluminum beams. The unit is located in front of the camera against the running direction of the conveyor belt. Relative to the conveyor belt, the bar light is located at a height of 118 mm and 140 mm for the lower and upper edges of the beam, respectively. Relative to the workpiece, the bar light is at a height of 70 mm and 92 mm for the lower and upper edges of the beam, respectively. The inclination angle is 30° to the horizontal. In addition, it was decided not to use the moving platforms of the initial setup, as this could lead to potential collisions with other setups. Instead, the samples will be placed directly on the conveyor belt in the final setup. Figure 6.3 outlines the setup. The sensor of the camera is located at a height of 300 mm relative to the workpiece surface. The line scan camera is powered via PoE by using an adaptor connected to a

power supply. The camera is connected to the hardware via ethernet. A Category 6 cable is used to ensure transmission of information according to requirements.

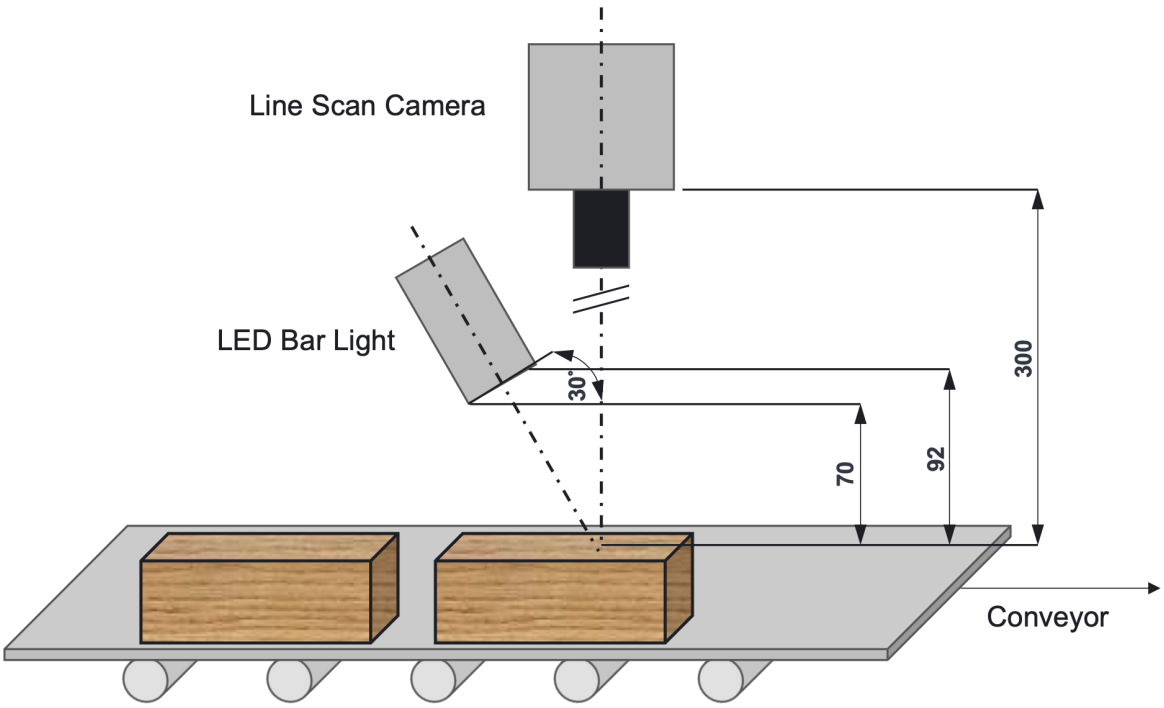


Figure 6.3: The setup of the camera system and the LED bar light above the conveyor system. The samples move from left to right at a constant speed.

6.3 Image Acquisition

In the following, the sampling process of the use case is described. First, information about the materials used as well as the practical implementation of the sampling process is given. Second, the calibration of the camera system is explained based on the available information. Third, the execution of the image acquisition process is described. Last, the processing steps applied to the collected data are explained.

6.3.1 Materials and Preparation

For the acquisition of the image data 380 low grade squared timbers made of spruce wood are used. The squared lumber is cut to size. The dimension of the samples used are [120 mm × 48 mm × 48 mm]($l \times w \times h$). To ensure that the pieces are evenly aligned, they are positioned

at equal intervals along the right-hand boundary of the conveyor as shown in figure 6.4. The positions for placing the individual samples are identified by markings on the conveyor. In addition, each specimen is clearly marked in advance so that its position and orientation can be determined.

Calibration

A Windows 10 notebook with Omron Sentech SDK (v1.2.1) is used for acquiring and storing the sampling image data. The calibration of the camera is done via the SDK software. The resolution in the X-direction of the line scan camera is reduced from 2048 to 704 pixels. This reduction is performed to reduce the background portion of the resulting images. Due to deviations, a width of the objects of 50 mm is assumed. The resolution of the image in mm/pixel is determined by:

$$\frac{\text{Width of object in mm}}{\text{Width of sensor in pixels}} = \frac{50}{704} = 0.071 \text{ mm}$$

With this information, the line rate is determined. Because of the dual line sensor of the camera, the result must be halved:

$$\frac{\text{Speed of conveyor in mm/s}}{\text{Size of pixel in mm}} = \frac{110}{0.071 \times 2} = 775 \text{ Hz}$$

The line period is then determined by inverting the line rate:

$$\frac{1}{\text{Line rate}} = \frac{1}{775} = 1290 \text{ µs}$$

The line period minus the camera specific line integration time gives the maximum exposure time. This parameter, however, is set automatically by the camera software to about 1288 µs. An overview of the calibration data used for sampling can be found in table 6.1.

Table 6.1: Calibration settings used for the line scan camera.

Parameter	Setting
Dimensions (W×H)	704×1924
X-Offset	720
Acquisition Frame Rate	0.81 Hz
Acquisition Line Rate	775.04 Hz
Exposure Time	1288.11 µs
Capture Mode	Multi Frame
Acquisition Frame Count	3

6.3.2 Sampling Process

Three samples at a time are manually placed on the conveyor system and then scanned by the camera in one pass (see figure 6.4). This means, three surfaces of the wooden cuboids are scanned by the system per run. Since there is no usable motion sensor technology in the Cyber-Physical Learning Factory at the time of data collection, the acquisition start of the camera is triggered by software. For this purpose, the conveyor belt is started simultaneously with the acquisition of the line scan camera.

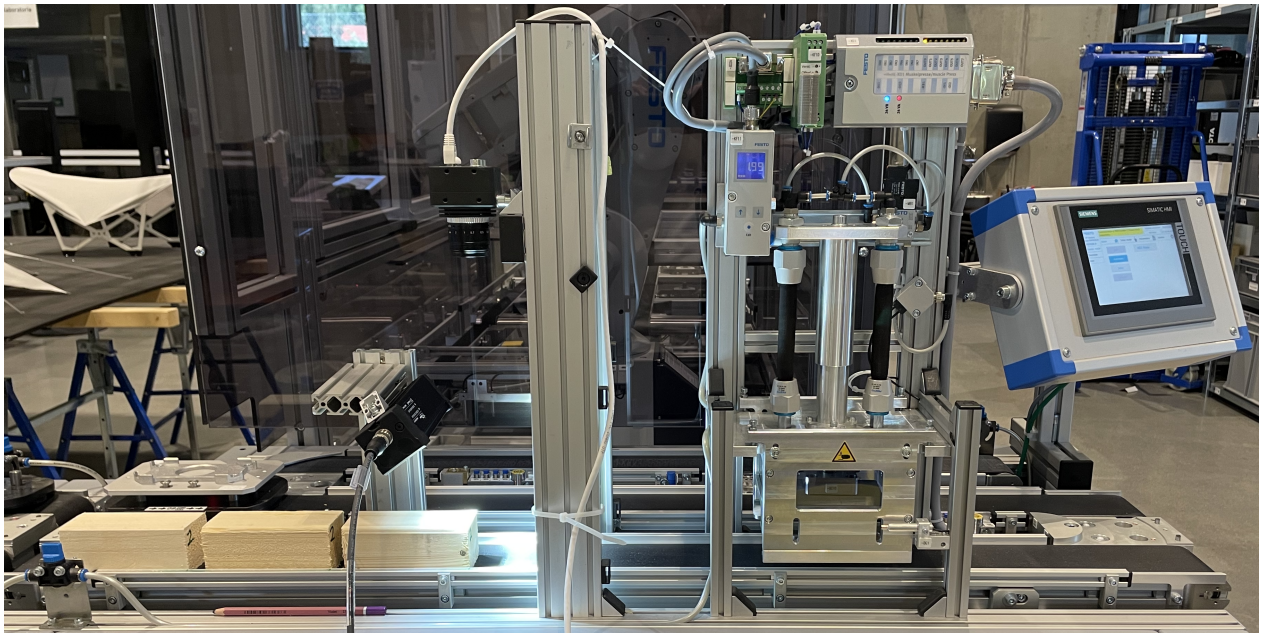


Figure 6.4: Start of a sampling process. Three samples at a time are placed on the conveyor belt and then scanned.

After scanning three samples, the image acquisition of the camera stops automatically. New samples are then placed on the conveyor and the acquisition process is repeated. The previously scanned samples are then conveyed to a discharge container. New samples are then placed on the conveyor belt and scanned. If problems occur during acquisition, e.g. due to collisions of the material with the conveyor or other installed equipment, the scans are repeated. After scanning all 380 samples, they are rotated by 90° and run through the process again. In total, the process is carried out four times, so that all samples are scanned once each from all four rectangular sides. The square side surfaces orthogonal to the fiber course are not scanned.

6.4 Acquired Data from Sampling

With the sampling process performed, a total of 1525 images of spruce wood surfaces are generated. Due to the manual activation of the conveyor belt and camera acquisition, there are some shifts of the subject within the image. Figure 6.5 showcases examples of the images acquired. To be analyzed with computer vision methods, the data is subsequently labeled and preprocessed.

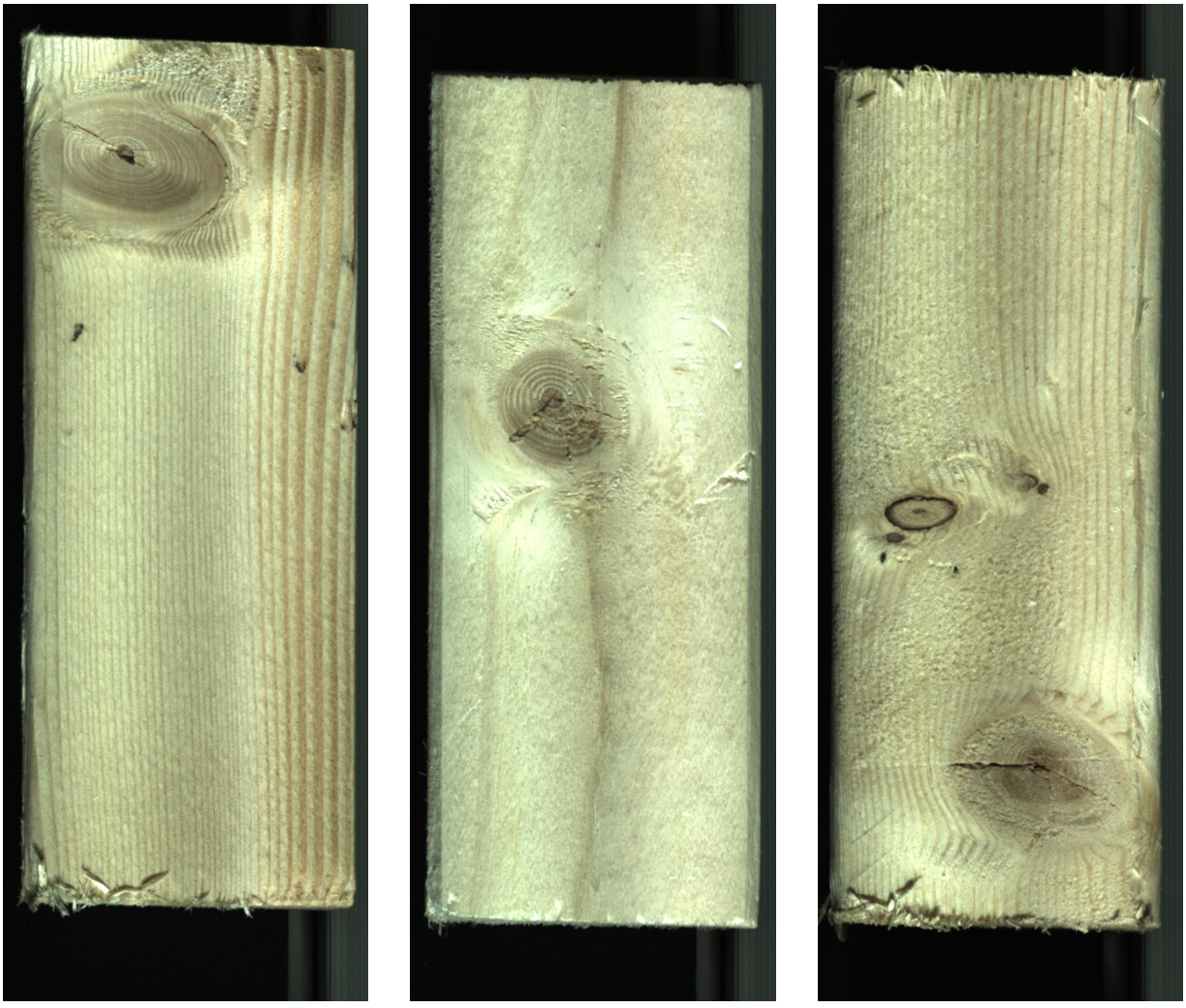


Figure 6.5: Example of image samples acquired during data acquisition.

6.4.1 Data Labeling

The Roboflow Annotate tool from the Computer Vision Platform Roboflow (Dwyer et al. 2022) is used for labeling the image files. The image files are uploaded to the suite and manually examined one by one to identify defects. Subsequently, if defects are present, rectangular bounding

boxes are drawn by hand to outline the present defects. Images without any defects are *marked null*, i.e. categorized as background class with an empty label file. To aid the labeling process, a model is pre-trained on parts of the data set from (Kodytek Pavel et al. 2021). This enables the use of the Label Assist tool, which suggests bounding box placements and their corresponding classes if a defect is detected by the model and a certain, pre-defined confidence threshold is met. Any defect bounding boxes incorrectly detected by the Label Assist tool are manually modified. In case of false positives, the bounding boxes are removed. In the case of incorrect annotation, the class of the bounding box is changed to the actual class. If the proposed bounding box has incorrect dimensions and thus only covers parts of the defect or larger parts of the background, it is be manually adjusted in its dimensions to fit the defect. The available classes align with the categories presented in section 5.1.1. Result of the manual labeling process are shown in figure 6.6.

The images are not further manipulated before exporting them. The created labels are exported as one text file per image, matching the data set introduced in section 5.1.1. This makes them compatible with the corresponding code for processing. The YOLO format is used as the format for the bounding box coordinates.

6.4.2 Data Analysis

Table 6.2 shows the distribution of defect types in the acquired image samples. As with the original data set from 5.1, the data set is highly imbalanced. The label overgrown does not occur at all in the data set. Furthermore, the distribution of defect types has changed. Whereas in the original data set, live knot is the most prevalent defect, it is crack in the one acquired by sampling. It has to be noted that 792 images, more half of the data set, are background images, i.e. they are without any defect.

Figure 6.7(a) visualizes the distribution of defect types in the data set. In figure 6.7(b) it can be seen that the number of samples with the respective defect class has a different order than the label frequency. Although crack is the most represented label, there are more images that contain at least one of the defects dead knot or live knot than there are images that contain at least one crack. That is, if cracks occur on images, there are usually several of them present at once.

6.4.3 Data Preprocessing

Following the labeling process, the image data is preprocessed to be suitable for analysis using the methods presented in Chapter 5.

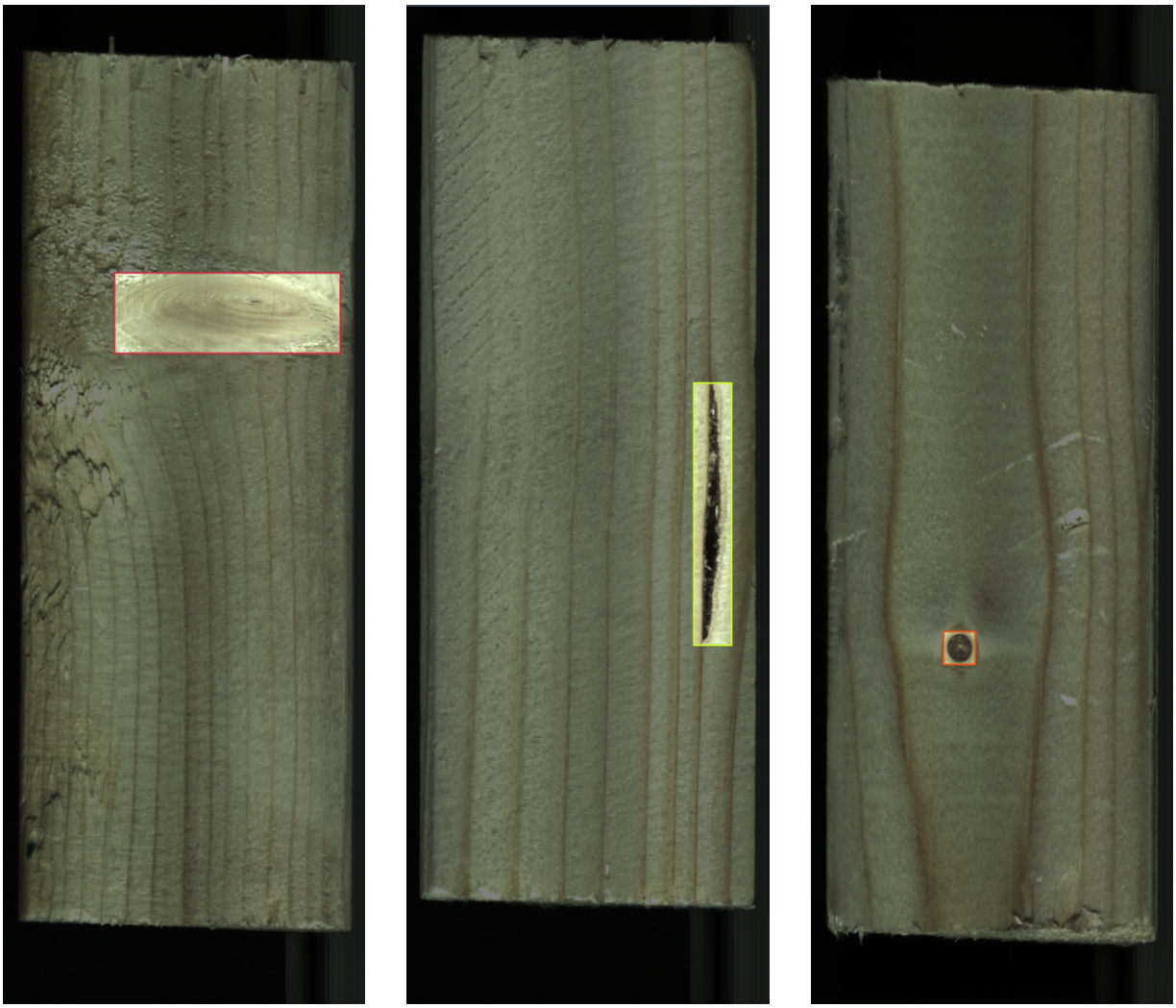
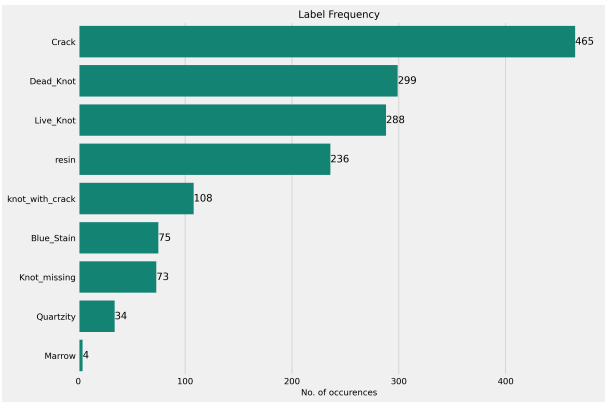


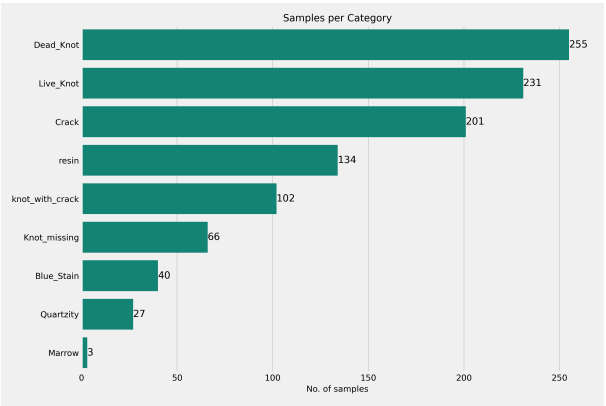
Figure 6.6: Acquired image samples after labeling in Roboflow Annotate. Rectangular bounding boxes are drawn to outline present defects. The colors of the bounding boxes are different based on the associated class label.

Table 6.2: Occurence of defects.

Defect class	No. of occurences	Percentage occurence	No. of samples with defect	Percentage of total samples
Crack	465	29.39 %	201	13.18 %
Dead knot	299	18.90 %	255	16.27 %
Live knot	288	18.20 %	231	15.15 %
Resin	236	14.92 %	134	8.79 %
Knot with crack	108	6.83 %	102	6.69 %
Blue stain	75	4.74 %	40	2.62 %
Knot missing	73	4.61 %	66	4.33 %
Quartzity	34	2.15 %	27	1.77 %
Marrow	4	0.25 %	3	0.20 %
Sum	1582	100 %	1059	



(a)



(b)

Figure 6.7: Bar plots representing the distribution of defect labels from the sampled images. Figure (a) displays the label frequency by defect label in descending order. Figure (b) presents the number of samples by defect label in descending order. The background class is excluded.

Rotation and Resizing

After labeling, the image data are in portrait format. To match the format of the data from section 5.1.1 and to be suitable for input into the classification networks, all images are rotated 90° to the right. This is done with the help of a program that simultaneously transforms the associated bounding box coordinates. Then the rotated images are scaled to a resolution of 700×256. Since the bounding box coordinates are in relative format, no further adjustment is required here.

Offline Data Augmentation

To investigate the impact of offline data augmentation on the use case data set, the same augmentation techniques as in 5.1.3 are applied to the data set. As with the data from 5.1, the methods are applied to the training as well as the validation data, while the test data set remains unchanged. Since the data set as a whole is small, a positive impact of offline data augmentation on training is expected.

Cleaning

Due to the fact that the defect marrow is only present four times in the dataset and only on three samples (see table 6.2 and figure 6.7), it is removed from the label space of the data set. Consequently, the integer IDs assigned to the classes in the label files are adjusted, which is necessary for the proper execution of the training process.

Organization

Following the procedure for the original data set (see 5.1.3), the sampling data are split in a distribution of 60 %-20 %-20 % between training, validation and test data set and distributed into corresponding folders and subfolders.

Table 6.3: Composition of raw use case data set.

Class label	Train	Val	Test
Blue stain	53	15	7
Crack	297	60	108
Dead knot	179	59	61
Knot missing	44	16	13
Live knot	169	68	51
Quartzity	17	13	4
Knot with crack	71	19	18
Resin	149	38	49
Background	470	153	169

Table 6.4: Composition of augmented use case data set.

Class label	Train	Val	Test
Blue stain	318	90	7
Crack	1782	360	108
Dead knot	1074	354	61
Knot missing	264	96	13
Live knot	1014	408	51
Quartzity	102	78	4
Knot with crack	426	114	18
Resin	894	228	49
Background	2820	918	169

Chapter 7

Results and Analysis

In this section the results of the analysis with regard to the research questions are presented. The section is structured according to the computer vision task and subsequently regarding the underlying data set. First, addressing research question 1a, the results of the multi-label image classification via CNNs are provided with regard to the general case and the use case (7.1 and 7.2). Second, the results of the object detection algorithms trained on the public data set are provided (research question 1b). The effects of offline data augmentation and transfer learning on object detection follow in subsections 7.3.2 and 7.3.3 (research questions 2 and 3a). Finally, the results of the object detection frameworks trained on the use case data are presented in a similar way.

7.1 Classification Results on Public Data Set

In this section, the results of multi-label classification of different CNN architectures are presented and compared. All models are trained on two NVIDIA-A100 with 80 GB of VRAM. The batch size is set to 32 per GPU and the models are trained for a maximum of 200 epochs. Details on the hyperparameters used are given in section 5.2. The models are trained on the training data set from section 5.1. The classes blue stain, knot missing and overgrown are not considered in this analysis and the corresponding class labels are removed from the label space in advance. As validation data during training, the validation data set presented is used. The test data set is used for the final evaluation of the models against unknown data. The data is tested against the test set specified in section 5.1. The test set is divided into batches of size 64 before evaluation. The performance metrics used in this analysis are the macro-averaged area under the ROC curve (AUC), macro-averaged area under the precision-recall curve (PRC), macro-averaged precision (Macro P), macro-averaged recall (Macro R), and macro-averaged F1 score (Macro F1).

Regarding run-time metrics, the models are tested in terms of their inference time (Inf_{batch}). As the test data is organized in batches before inference, the inference time refers to the time required to process one batch.

Table 7.1 lists the results of the ResNet and EfficientNet models trained on the original data set. The smallest model in terms of number of trainable parameters, ResNet-18, achieves the highest values across all performance categories. At the same time, the model takes the longest for inference with an inference time of 267 ms. In terms of AUC, all models except EfficientNetV2-S realize scores of 90 % or more. PRC values of more than 80 % are achieved by both of the ResNet models, while EfficientNetV2-S achieves 50.4 %. A similar picture emerges for macro-averaged precision, recall, and F1 score, where the scores of EfficientNet-V2s are lower by a large margin compared to the other models.

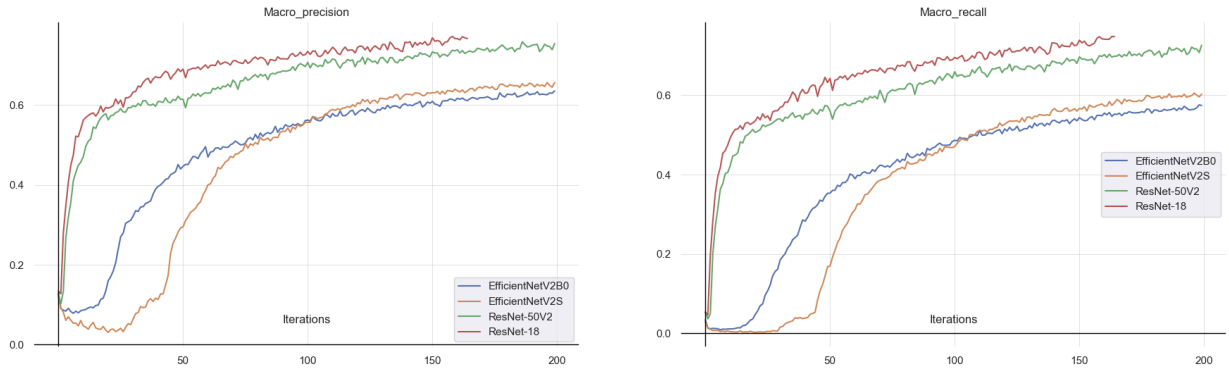
Table 7.1: Test results of ResNet and EfficientNet models. The models were trained on the original data set. The best in-class performance results are marked bold.

Model	#Param.	Inf_{batch}	AUC	PRC	Macro P	Macro R	Macro F1
ResNet-18	22.7M	267 ms	**95.0 %**	**83.7 %**	**72.7 %**	**71.2 %**	**70.0 %**
ResNet-50V2	69.7M	**95 ms**	93.3 %	81.5 %	72.0 %	66.3 %	65.6 %
EfficientNetV2-B0	34.7M	102 ms	90.5 %	72.4 %	64.2 %	54.9 %	55.4 %
EfficientNetV2-S	49.0M	127 ms	77.4 %	50.4 %	41.2 %	31.2 %	29.4 %

In terms of training performance, the ResNet models and the EfficientNet models exhibit distinct behavior. As figure 7.1(a) shows, the two ResNet models demonstrate a steep upward rise in macro precision from the initial epochs. After that the rate of improvement lessens. In contrast, the EfficentNet models improve in performance first after several epochs. In the beginning, there is even a decrease, which in the case of EfficientNetV2-S lasts for about 25 epochs. A similar picture emerges for the development of the macro-recall as seen in Figure 7.1(b). Here, the ResNet models also increase steeply initially, while the EfficientNet models remain close to zero for several epochs. Moreover, in both cases, the curve of EfficientNetV2-S surpasses that of the -B0 model after about 100 epochs.

7.2 Classification Results on Use Case Data Set

This section addresses the results of the CNN models for which were trained on the image data collected for the use case. In contrast to section 7.1, the models in this section are trained for a maximum of 500 epochs. Information about the set of hyperparameters is provided in section 5.2. The test partition of the raw data set introduced in section 6.4 is used for validation. The



(a) (b)

Figure 7.1: Development of (a) macro-averaged precision and (b) macro-averaged recall of the CNN models during training.

evaluation of the models against the test data is performed with a batch size of 64 as in section 7.1.

The results for the CNN models validated against the use case test data set are listed in Table 7.2. ResNet-50V2 achieves the second highest values with a significant distance to the scores of ResNet-18. It is noteworthy to mention that both EfficientNet models report 0 % for macro-averaged precision, recall and F1 score.

Table 7.2: Test results of ResNet and EfficientNet models. The models were trained on the use case data set. The best in-class performance results are marked bold.

Model	#Param.	Inf_{batch}	AUC	PRC	Macro P	Macro R	Macro F1
ResNet-18	22.7M	**132 ms**	**67.0 %**	**24.7 %**	**21.1 %**	**9.6 %**	**11.6 %**
ResNet-50V2	69.7M	183 ms	52.5 %	17.3 %	3.0 %	1.2 %	1.5 %
EfficientNetV2-B0	34.7M	149 ms	50.2 %	13.4 %	0 %	0 %	0 %
EfficientNetV2-S	49.0M	203 ms	49.8 %	14.2 %	0 %	0 %	0 %

7.3 Detection Results on Public Data Set

In this section, the benchmark results of the object detection models trained or fine-tuned on the public data set from section 5.1 are provided. Section 3.5 provides an overview and explanations about the metrics used. The baseline results by training from scratch are communicated in section 7.3.1. Sections 7.3.2 and 7.3.3 show the results for training on offline augmented data and transfer learning, respectively as well as their comparison to the baseline results.

All models in this section are trained and evaluated on the same hardware, as specified in section 5.2. To evaluate the models, all models are tested against the same test data set, which consists solely of instances unknown to the model. For this purpose, the models are using their best stored weights. All metrics except mAP_{50}^{val} are determined via the test set. The mAP_{50}^{val} values are determined using the validation data set used by the respective model during training. In the following, if not explicitly stated, all metrics refer to those determined with the test set. The metrics P^{test} and R^{test} refer to the unweighted mean precision and unweighted mean recall across all classes. The number of model parameters and FLOPs is extracted directly from the model's console output during training.

7.3.1 Baseline Results

In this section, the results of the object detection algorithms by training from scratch are presented. Table 7.3 shows the results of the models trained on the original data set (see 5.1). It can be seen that all models except YOLOv7 achieve mAP_{50} scores of over 70 %. YOLOv8-M achieves the best results for mAP_{50}, mAP_{50-95}, and R^{test} with values of 76.4 %, 45.4 %, and 70.7 %, respectively. In terms of P^{test} it is outperformed by YOLOv8-S by a margin of 2.1 %. The fastest model in the test is YOLOv7-tiny (YOLOv7-T) with a detection speed of 227 FPS. It is interesting to see that YOLOv8-L performs worse than the smaller -M model in all aspects. The base YOLOv7 model has the lowest values overall and is outperformed in all metrics.

Table 7.3: Test results of YOLOv8 and YOLOv7 models in different scales. The models were trained on the original data set. The best in-class performance results are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	3.0M	8.1G	196	74.2 % / 74.3 %	44.4 %	72.7 %	67.5 %
YOLOv8-S	11.1M	28.5G	208	74.5 % / 75.6 %	43.8 %	**77.4 %**	67.0 %
YOLOv8-M	25.8M	78.7G	169	**76.4 %** / 75.7 %	**45.4 %**	75.3 %	**70.7 %**
YOLOv8-L	43.6M	164.9G	141	75.5 % / 76.0 %	44.7 %	74.5 %	70.6 %
YOLOv7-T	6.0M	13.1G	**227**	74.1 % / 73.4 %	41.5 %	73.8 %	69.5 %
YOLOv7	36.5M	103.3G	139	66.2 % / 65.7 %	36.9 %	70.4 %	63.2 %

Per Class Results

The models are now compared on a per-class basis. It should be noted here that in this section, mean precision should not be confused with the mAP_{50} metric. Instead, it is the unweighted average of all the per-class precision values. For reasons of clarity, the class names in the tables

are abbreviated as follows: Blue stain (BS), crack (CR), dead knot (DK), knot missing (KM), live knot (LK), marrow (MA), quartzity (QU), knot with crack (KC), and resin (RE).

The per-class results for precision are listed in Table 7.4. Here it can be seen that the YOLOv8-S model achieves the highest precision value in seven of the nine classes, and the overall highest mean precision. It is noticeable that YOLOv7 and YOLOv7-tiny achieve values of 100 % and 97.6 % for the class BS, which is significantly higher than all other models. The class with the highest precision averaged accross all models is MA with 82.95 %, followed by RE with 82.18 %. QU (67.28 %) has the lowest precision on average, followed by CR (67.97 %).

Table 7.4: Per class precision values.

Model	Mean	BS	CR	DK	KM	LK	MA	QU	KC	RE
v8-N	72.7 %	46.1 %	72.4 %	82.0 %	70.4 %	80.2 %	85.3 %	65.2 %	69.5 %	82.8 %
v8-S	**77.4 %**	55.4 %	**74.5 %**	**84.1 %**	**78.7 %**	**84.0 %**	**87.2 %**	**75.5 %**	71.5 %	**85.9 %**
v8-M	75.3 %	60.4 %	67.8 %	81.7 %	77.0 %	83.1 %	83.9 %	67.4 %	**73.4 %**	82.7 %
v8-L	74.5 %	57.9 %	70.1 %	80.0 %	75.1 %	82.4 %	84.7 %	67.4 %	71.0 %	81.5 %
v7-T	73.8 %	97.6 %	62.8 %	75.8 %	61.1 %	71.7 %	80.5 %	69.2 %	64.9 %	80.8 %
v7	70.4 %	**100 %**	60.2 %	70.5 %	65.9 %	60.9 %	76.1 %	59.0 %	61.2 %	79.4 %

The per-class values for recall are showcased in table 7.5. YOLOv8-M achieves the highest recall values in five of nine classes and the highest mean recall overall. YOLOv7-tiny and YOLOv7 each achieve best per-class scores in two of nine classes. A noteworthy observation is that YOLOv7 demonstrates a recall of 0 % for the blue stain class. The classes with the highest average recall across models are MA and LK with 90.35 % and 84.23 %, respectively. The classes with the lowest average recall are BS and QU with 32.68 % and 36.68 %, respectively.

Table 7.5: Per class recall values.

Model	Mean	BS	CR	DK	KM	LK	MA	QU	KC	RE
v8-N	67.5 %	33.3 %	66.4 %	77.6 %	64.1 %	84.4 %	88.7 %	41.5 %	78.5 %	73.2 %
v8-S	67.0 %	38.1 %	66.7 %	75.6 %	67.9 %	78.9 %	88.7 %	36.2 %	78.5 %	72.6 %
v8-M	**70.7 %**	**43.7 %**	**72.1 %**	78.5 %	**68.8 %**	81.7 %	87.9 %	**47.3 %**	80.0 %	**76.6 %**
v8-L	70.6 %	42.9 %	69.8 %	81.0 %	65.7 %	82.5 %	90.5 %	44.4 %	82.3 %	76.4 %
v7-T	69.5 %	38.1 %	66.9 %	83.0 %	68.5 %	88.2 %	**93.4 %**	29.0 %	**83.1 %**	75.7 %
v7	63.2 %	0 %	68.7 %	**84.2 %**	62.1 %	**89.7 %**	92.9 %	21.7 %	81.4 %	68.3 %

7.3.2 Effects of Offline Data Augmentation

In the following, the results of the validation of the models trained on the offline-augmented data according to section 5.1.3 are presented. The training is performed under the same condi-

tions as training from scratch, that is, only the underlying data sets are changed. The validation results on the test set are based on the same data set and procedure as before. However, to determine mAP_{50}^{val} in this analysis, the models are tested against the augmented validation set. Table 7.6 provides an overview about the test results of the different models.

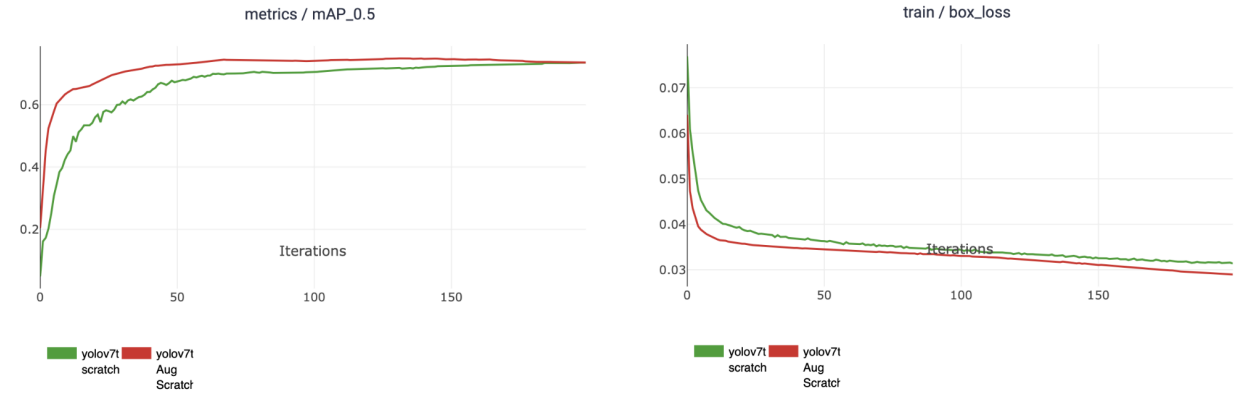
Table 7.6: Test results for training on offline-augmented public data set. The best per-category performance metrics are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N (AUG)	3.0M	8.1G	145	72.5 % / 72.7 %	43.2 %	71.4 %	66.4 %
YOLOv8-S (AUG)	11.1M	28.5G	149	73.8 % / 72.4 %	44.4 %	74.7 %	68.4 %
YOLOv8-M (AUG)	25.8M	78.7G	112	75.3 % / 74.8 %	46.3 %	**78.8 %**	68.4 %
YOLOv8-L (AUG)	43.6M	164.9G	104	76.9 % / 76.2 %	**46.5 %**	75.5 %	70.7 %
YOLOv7-T (AUG)	6.0M	13.1G	**200**	**77.2 %** / 74.9 %	45.3 %	72.4 %	**72.5 %**
YOLOv7 (AUG)	36.5M	103.3G	130	76.9 % / 76.4 %	46.3 %	76.9 %	70.2 %

Effects on Model Training

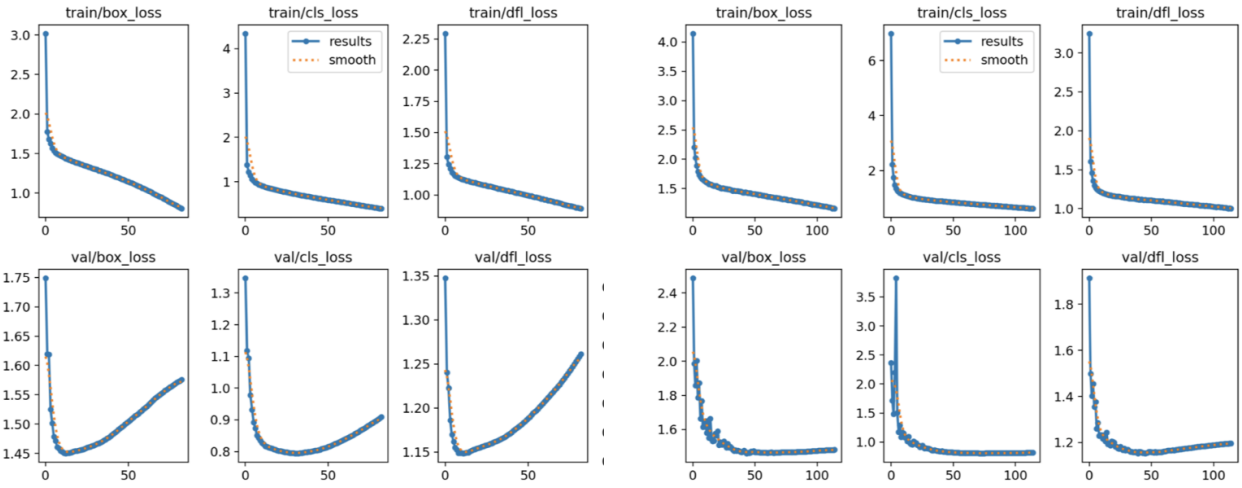
Offline augmentation affects the training of the models. Since the data set was multiplied in size due to offline data augmentation, significant increases in training time are observed. For instance, when training on the augmented data set, YOLOv7 demands more than 69 hours using an A100 40GB, whereas training on the raw data set completes in approximately 13 hours. In addition to impacts on training time, changes in training performance can be observed. As shown in figure 7.2, using YOLOv7-tiny as an example, there are noticeable changes in convergence speed and training loss of the models . In 7.2(a), it can be seen that the augmented model gains in mAP_{50} faster than the non-augmented model in early epochs. However, eventually, both models converge at the end of training. Figure 7.2(b) demonstrates differences regarding training loss. The training box loss of the augmented model drops quicker and reaches a lower minimum. These changes can be observed in all models examined.

As shown in figure 7.3, differences in the behavior of the loss can be observed between the models trained on augmented data and those trained on the original data. Subfigure 7.3(a)(bottom) displays the training losses of the augmented model consistently decreasing, while its validation losses reach a minimum after a short time and then increase steadily. Subfigure 7.3(b) depicts the behavior of the baseline model. There, validation box loss and dfl loss exhibit a light increase over an extended period. Another noteworthy observation is that when trained on the offline-augmented data set, all YOLOv8 models demonstrate an earlier activation of the early stopping criterion in comparison to their training on the raw data set.



(a) (b)

Figure 7.2: Effects of-offline augmentation on the training process of the YOLOv7-tiny models in terms of (a) mAP_{50}, and (b) training box loss.



(a) (b)

Figure 7.3: Comparison of training (top) and validation (bottom) losses over epochs for the YOLOv8-L model. Subfigure (a) presents the loss curves of the model trained on original data. Subfigure (b) displays the corresponding loss curves for the model trained on augmented data.

Effects on Model Performance

Table 7.7 shows the test performance differences of the models trained on the augmented data relative to the baseline models. It is noticeable that all models have a slower processing time. Regarding the performance metrics, no clear picture emerges. YOLOv8-N, -S and -M decrease in mAP_{50}, while YOLOv8-L, YOLOv7-tiny, and YOLOv7 increase. Regarding mAP_{50-95}, all models achieved higher mAP_{50-95} values with the exception of the YOLOv8-N model. For YOLOv8-L, and YOLOv7, the training on offline augmented data improves all performance metrics except FPS. YOLOv7 demonstrates the highest increases. In contrast, YOLOv8-N shows a deterioration in all areas.

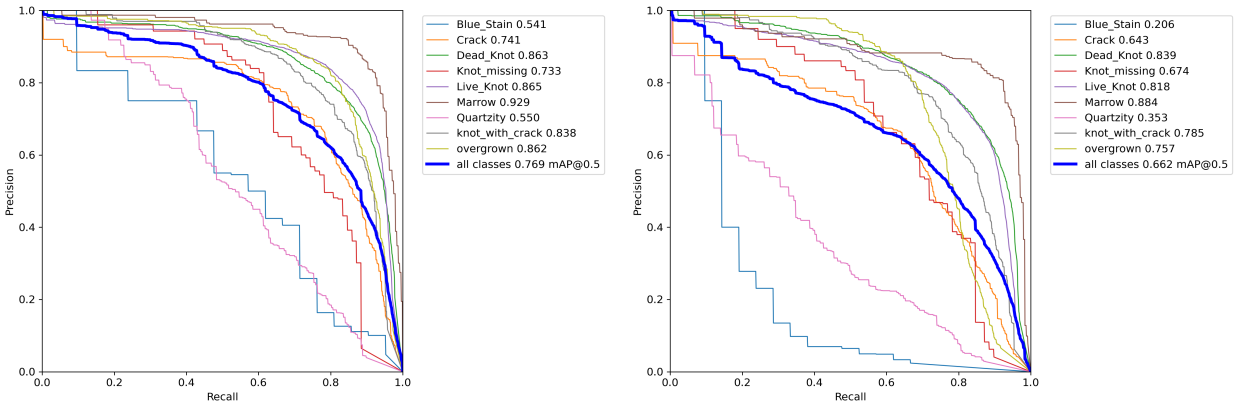
Table 7.7: Relative change in performance when trained on augmented data compared to training from scratch.

Model	FPS	mAP_{50}^{test}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	-51	-1.7 %	-1.2 %	-1.3 %	-1.1 %
YOLOv8-S	-59	-0.7 %	+0.6 %	-2.7 %	+1.4 %
YOLOv8-M	-57	-1.1 %	+0.9 %	+3.5 %	-2.3 %
YOLOv8-L	-37	+1.4 %	+1.8 %	+1.0 %	+0,1 %
YOLOv7-T	-27	+3.1 %	+3.8 %	-1.4 %	+3,0 %
YOLOv7	-9	+10.7 %	+9.4 %	+6.5 %	+7.0 %

The change in the performance metrics of YOLOv7 can be seen in the comparison of the precision-recall (PR) curves of the models. Figure 7.4 (a) and (b) display the test PR curves of the model trained on augmented data and the model trained on raw data, respectively. The curve for blue stain demonstrates a noticeable increase in AUC. Also visible is an increase in mean precision at higher confidence thresholds.

7.3.3 Effects of Transfer Learning

In the following, the effects of transfer learning via pre-training on a large image database are presented. All models are initialized with weights from corresponding models pre-trained on the MS COCO data set. Other training parameters remain unchanged and correspond to the settings from sections 7.3.1 and 7.3.2. A freezing of layers of the models is not applied. An overview about the acquired results is given in table 7.8.



(a) (b)

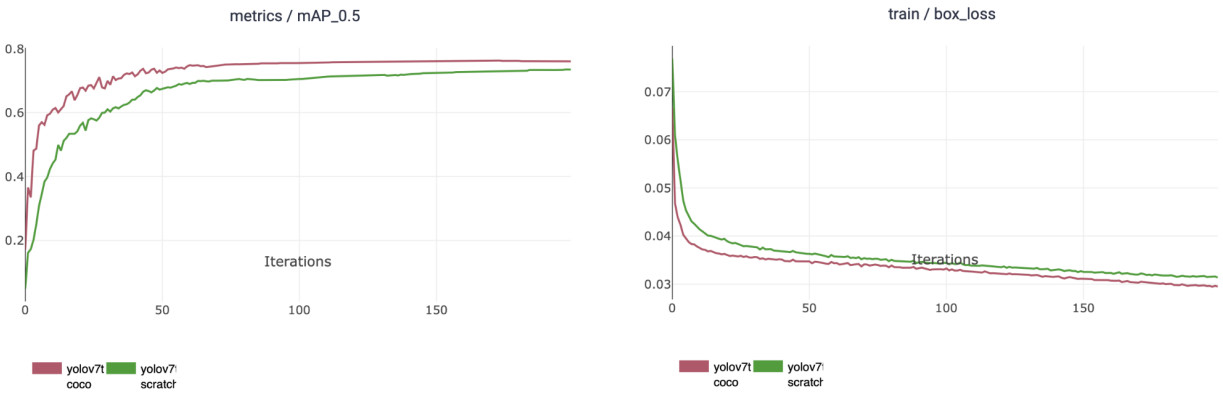
Figure 7.4: Comparison of precision-recall-curves between the YOLOv7 model trained (a) on augmented data and (b) on raw data. A noticeable increase in AUC for blue stain can be observed.

Table 7.8: Test results for training on public data set with transfer learning. The best per-category performance metrics are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	3.0M	8.1G	204	73.9 % / 75.8 %	45.5 %	69.2 %	**72.8 %**
YOLOv8-S	11.1M	28.5G	204	75.6 % / 75.6 %	46.0 %	75.7 %	70.3 %
YOLOv8-M	25.8M	78.7G	169	75.9 % / 76.4 %	46.1 %	72.0 %	71.2 %
YOLOv8-L	43.6M	164.9G	141	74.6 % / 76.9 %	**46.3 %**	**76.3 %**	69.7 %
YOLOv7-T	6.0M	13.1G	**217**	**76.2 %** / 76.4 %	45.0 %	74.7 %	71.4 %
YOLOv7	36.5M	103.3G	139	59.7 % / 60.7 %	30.7 %	70.4 %	58.3 %

Effects on Model Training

The training behavior between the pre-trained models and those trained from scratch differs. With the exception of YOLOv7, the pre-trained models all demonstrate a more rapid increase in their metrics during training in conjunction with lower training and validation loss at the beginning of the training. Figure 7.5 demonstrates this behavior using YOLOv7-tiny as an example.



(a) (b)

Figure 7.5: Effects of transfer learning with pre-trained weights on MS COCO on the training process of the YOLOv7-tiny models in terms of (a) mAP_{50}, and (b) training box loss. The green graph depicts the model trained from scratch. The pre-trained results are visualized in red.

Effects on Model Performance

Table 7.9 shows the changes in test performance of the pre-trained models compared to the baseline models. YOLOv8-S and YOLOv7-T show increases in mAP_{50} by +1.1 % and +2.1 %, respectively. The value decreases for the other models, with YOLOv7 showing the largest decrease with -6.5 %. Increases in mAP_{50-95} are achieved by all the pre-trained models except YOLOv7. YOLOv8-N, -S, and -M achieve lower mean precision values and higher mean recall values compared to their baseline models. The pre-trained YOLOv8-L variant reaches a higher mean precision and a lower mean recall compared to the baseline model. Pre-trained YOLOv7-tiny shows increases in all metrics except processing speed. YOLOv7 metrics decrease or do not exhibit any change. It should be noted, however, that the processing speed is subject to fluctuations and can therefore vary slightly between different test runs.

Table 7.9: Relative change in performance when trained using transfer learning compared to training from scratch.

Model	FPS	mAP_{50}^{test}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	+8	-0.3 %	+1.1 %	-3.5 %	+5.3 %
YOLOv8-S	-4	+1.1 %	+2.2 %	-1.7 %	+3.3 %
YOLOv8-M	-	-0.5 %	+0.7 %	-3.3 %	+0.5 %
YOLOv8-L	-	-0.9 %	+1.6 %	+1.8 %	-0,9 %
YOLOv7-T	-10	+2.1 %	+3,5 %	+0,9 %	+1,9 %
YOLOv7	-	-6.5 %	-6.2 %	-	-4.9 %

7.4 Detection Results on Use Case Data Set

This section summarizes the test results of the models trained on the use case data. This section is similar to the structure of section 7.3. First, the baseline results, i.e. by training from scratch are presented. This is followed by the effects of offline data augmentation on training and model performance. Finally, the results of transfer learning and the effects on training and model performance are provided.

The general procedure for the experiments on the use case data set corresponds to the procedure of section 7.3. There are no differences in the hardware and software used. However, in contrast to the training on the public data set, the number of maximum epochs is increased to 300. Otherwise, no changes are made to the set of hyperparameters.

7.4.1 Baseline Results

Table 7.10 lists the results from the models trained from scratch on the raw use case data set (section 6.4). The values of the performance metrics are significantly when using the use case data for training compared to using the public data set. The highest mAP_{50} is achieved by YOLOv8-M with 30.4 %. All other models results in this category are below 30 %. Noticeable are the differences between mAP_{50}^{test} and mAP_{50}^{val}. The score determined via the validation set is higher than the results via the test set by a significant margin. This observation applies to all models examined. In terms of mean precision, YOLOv7 scores significantly higher than the other models with a value of 53.9 %. Another interesting observation is that the process speed is significantly lower than that of the models trained on the public data set for all models, despite the same resolution of the samples.

Table 7.10: Test results of YOLOv8 and YOLOv7 models in different scales. The models are trained on the data set from the use case. The best in-class performance results are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	3.0M	8.1G	139	23.0 % / 32.4 %	10.3 %	32.3 %	25.6 %
YOLOv8-S	11.1M	28.5G	**154**	27.5 % / 35.4 %	**14.2 %**	27.1 %	38.0 %
YOLOv8-M	25.8M	78.7G	125	**30.4 %** / 33.6 %	13.0 %	25.6 %	**39.4 %**
YOLOv8-L	43.6M	164.9G	108	27.2 % / 35.4 %	14.1 %	32.9 %	29.3 %
YOLOv7-T	6.0M	13.1G	149	26.5 % / 33.6 %	12.2 %	40.8 %	31.0 %
YOLOv7	36.5M	103.3G	120	27.6 % / 35.3 %	13.4 %	**53.9 %**	30.1 %

7.4.2 Effects of Offline Data Augmentation

In the following, the test results of models trained on the use case data that has undergone offline data augmentation are presented. Table 7.11 lists the results for all models examined. In contrast to the baseline models, the training is performed on a data set enlarged by a factor of six. The size of the validation set is correspondingly increased. This results in training and validation data sets of 5490 and 1830 images, respectively. The test set remains the same size with 305 images.

Table 7.11: Test results for training on offline-augmented use case data set. The best per-category performance metrics are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	3.0M	8.1G	152	23.7 % / 30.3 %	12.5 %	25.1 %	32.7 %
YOLOv8-S	11.1M	28.5G	154	27.3 % / 34.8 %	14.5 %	43.8 %	30.7 %
YOLOv8-M	25.8M	78.7G	130	27.2 % / 35.6 %	**15.2 %**	33.1 %	31.6 %
YOLOv8-L	43.6M	164.9G	109	27.2 % / 37.4 %	14.9 %	34.5 %	31.6 %
YOLOv7-T	6.0M	13.1G	**161**	27.4 % / 37.3 %	14.0 %	**45.0 %**	32.5 %
YOLOv7	36.5M	103.3G	118	**28.3 %** / 40.1 %	14.2 %	31.1 %	**36.0 %**

Effects on Model Performance

Similar to the public data set, there are changes in the performance of the models after training on the augmented use case data in comparison to training on the raw use case data.

Table 7.12 provides an overview of the differences in metrics for all models. Looking at mAP_{50}, both increases and decreases in performance are seen, with the largest increase being 0.9 percentage points. The mAP_{50-95} of all models improved after offline data augmentation with a

maximum increase of 2.2 percentage points. The changes in mean precision show a wide variation across the models with changes ranging from -22.8 % in the case of YOLOv7 to +16.7 % in the case of YOLOv8-S. The changes for mean recall range from -7.8 % to +5.9 %. For both mean precision and mean recall metrics, performance improvement was observed in four out of the six models.

Table 7.12: Relative change in performance when trained on augmented use case data compared to training from scratch.

Model	FPS	mAP_{50}^{test}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	+13	+0.7 %	+2.2 %	-7.2 %	+7.1 %
YOLOv8-S	-	-0.2 %	+0.3 %	+16.7 %	-7.3 %
YOLOv8-M	+5	-3.2 %	+2.2 %	+7.5 %	-7.8 %
YOLOv8-L	+1	-	+0.8 %	+1.6 %	+2,3 %
YOLOv7-T	+12	+0.9 %	+1.8 %	+4.2 %	+1,5 %
YOLOv7	-3	+0.7 %	+0.8 %	-22.8 %	+5.9 %

7.4.3 Effects of Transfer Learning

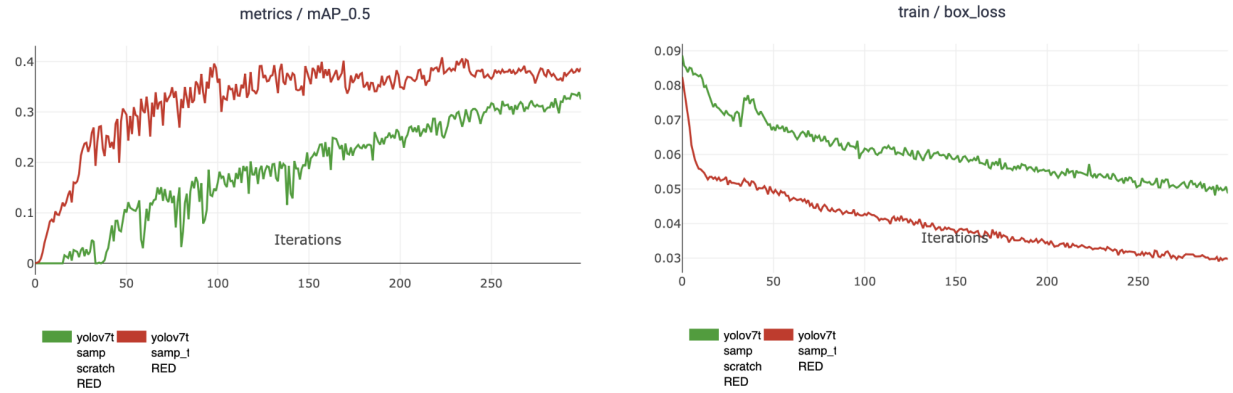
In this section, the results of transfer learning by fine-tuning pre-trained models using the data from the use case are outlined. Unlike the models trained on the public dataset, which are initially pre-trained on MS COCO, the models in this section are initialized with the optimized weights obtained from models trained on the offline augmented public dataset. Similiar to the previous study on transfer learning in object detection, the same set of hyperparameters is used for training from scratch is employed here. Moreover, all layers of the models remain trainable, i.e. no freezing of layers is implemented. Table 7.13 shows the results of the tests.

Table 7.13: Test results for training on use case data set with transfer learning. The best per-category performance metrics are marked bold.

Model	#Param.	FLOPs	FPS^{A100}	mAP_{50}^{test} / mAP_{50}^{val}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	3.0M	8.1G	145	24.8 % / 35.6 %	13.4 %	51.2 %	29.1 %
YOLOv8-S	11.1M	28.5G	149	29.0 % / 39.3 %	13.6 %	54.8 %	26.8 %
YOLOv8-M	25.8M	78.7G	127	30.3 % / 39.0 %	15.6 %	32.3 %	36.6 %
YOLOv8-L	43.6M	164.9G	106	**34.7 %** / 39.7 %	**18.4 %**	38.5 %	39.5 %
YOLOv7-T	6.0M	13.1G	**132**	26.7 % / 41.2 %	13.4 %	**47.5 %**	30.9 %
YOLOv7	36.5M	103.3G	122	24.0 % / 40.6 %	16.8%	38.0 %	**42.2 %**

Effects on Training

As with the public data set, transfer learning impacts the training process while training with the use case data. Initializing the models with the best weights of the models from section 7.6 leads to a faster improvement of the metrics of the models as well as lower initial and faster declining losses during training. This can be seen well in the training curves regarding mAP_{50} and box loss of the yolov7-tiny model (see figure 7.6). Also remarkable is that all YOLOv8 models stagnate or deteriorate in their performance after a maximum of 65 epochs, so that early stopping sets in. In particular, YOLOv8-S experiences no improvement after just 28 epochs. On the other hand, during training from scratch, early stopping sets in after 126 epochs in the earliest case (YOLOv8-M).



(a) (b)

Figure 7.6: Effects of transfer learning with weights from section 7.6 on the training process of the YOLOv7-tiny model in terms of (a) mAP_{50}, and (b) training box loss.

Effects on Model Performance

As seen in table 7.14, there are changes in model performance, when initializing the models with weights from models trained on a similar domain. The performance in terms of mAP_{50} is increased in for all models except YOLOv8-M. The highest increase in mAP_{50} is achieved by YOLOv8-L with +7.5 percentage points. YOLOv7 achieves a plus of 6.4 percentage points. The results for mAP_{50-95} are higher in all cases except YOLOv8-S. The change in mean precision ranges from 27.7 to -15.9 percentage points, with YOLOv8-S showing the largest increase and YOLOv7 the largest decrease. The change in mean recall also shows a high variance with changes ranging from -11.2 to +12.1 percentage points.

Overall, YOLOv8-N and -L are the only models in this comparison that improve in all performance metrics with respect to detection quality through transfer learning.

Table 7.14: Relative change in performance when trained on augmented use case data compared to training from scratch.

Model	FPS	mAP_{50}^{test}	mAP_{50-95}^{test}	P^{test}	R^{test}
YOLOv8-N	+6	+1.8 %	+3,1 %	+18.9 %	+3.5 %
YOLOv8-S	-5	+1.5 %	-0.6 %	+27.7 %	-11.2 %
YOLOv8-M	+2	-0.1 %	+2.6 %	+6.6 %	-2.8 %
YOLOv8-L	-1	+7.5 %	+4.3 %	+5.6 %	+10.2 %
YOLOv7-T	-18	+0.2 %	+1.2 %	+6.7 %	-0,1 %
YOLOv7	+1	+6.4 %	+3.4 %	-15.9 %	+12.1 %

Chapter 8

Discussion

8.1 Application of Computer Vision in Wood Defect Detection

8.1.1 Image Classification Models

Research question 1a. asked how suitable ResNet and EfficientNet models are for detecting defects on wood surfaces. In this thesis, the performance of two ResNet models (ResNet-18, ResNet-50V2) and two EfficientNet models (EfficientNetV2B0, EfficientNetV2-S) were evaluated. The evaluation was based on a publicly available data set as well as on data collected during the use case. The models were evaluated using the respective macro-averaged classification metrics: ROC AUC, PRC AUC, precision, recall, and F1 score. Additionally, the inference time of the models was determined when performing inference on unknown data.

Comparison of Model Performance

For the comparison of the architectures trained on the public data set, a clear picture emerges. ResNet-18 outperforms all other models with respect to the evaluated classification metrics. For the macro-averaged metrics ROC AUC and PRC AUC, calculated over a wide range of thresholds, values of up to 95 % and 83.7 % were obtained using ResNet-18. For the macro-averaged metrics precision, recall, and F1 score, values of 72.7 %, 71.2 %, and 70 % were achieved in this case. ResNet50-V2, EfficientNetV2-B0 and EfficientNetV2-S follow in descending order in terms of performance, with the difference between ResNet-18 and EfficientNetV2-S scores of more than 40 percentage points in terms of macro recall and precision. Hence, EfficientNet-V2S achieves scores of 77.4 % and 50.4 % for ROC AUC and PRC AUC, and scores of 41.2 %, 31.2 %, and 29.4 % for precision, recall, and the F1 score, respectively. In general, the two ResNet models perform

significantly better in terms of recall, precision, and the derived metrics.

This result was not to be expected, since ResNet-18 is the smallest (and oldest) model in this study. This contradicts the paradigm that deeper models usually lead to better performance (He et al. 2016*a*). Also contradictory are the comparatively poor results of the EfficientNet models, which perform equally or better than ResNet-50 in the authors' benchmarks (Tan & Le 2019, 2021). On the other hand, these benchmarks were performed based on the ImageNet challenge, which is not comparable to the procedure and underlying data in this work.

One reason for the comparatively good performance of ResNet-18 could be its low complexity combined with the small amount of training data. This combination could help ResNet-18 converge faster during training. The CNNs were each trained for only 200 epochs when training on the original data set. Due to the limited training time, the models could have been prevented from reaching a better or optimal solution. This argument is supported by the lack of onset of early stopping in the deeper models as well as the course of the training curves. In the curves it can be observed that the EfficientNet models demonstrate starting difficulties, but then improve continuously until the maximum number of epochs is reached. ResNet-18 on the other hand experiences early stopping. This leads to the assumption that the performance of the larger models would have improved further with longer training.

Remarkably, in the case of the original data, ResNet-18 is inferior to the other models in terms of inference speed. This is counterintuitive since ResNet-18 is the smallest and least complex of the models examined. Possible reasons could lie in the implementation, as the model is not imported directly from the TensorFlow Keras API and may not have been optimally optimized for it. On the other hand, external factors such as hardware utilization level or background processes may also have increased the inference time. This is supported by the fact that, in the case of inference on the use case data, the model is the fastest in comparison.

Interpreting the Results

To answer the question about the suitability of the models for defect detection in the wood manufacturing domain, it's crucial to examine the results and their significance. Therefore, the results of the models are now interpreted and contextualized within the existing research.

A high ROC AUC value indicates that the model can generally distinguish well between defective and non-defective instances. Besides EfficientNetV2-S, all models achieved values higher than 90 % in this category, indicating these models have a good discrimination ability. Precision measures how often the model predicts correctly when labeling images as defective. In the case of ResNet-18 this means that in 72.7 % of the cases where a defect is predicted, there is actually

a defect present. Recall represents the ratio of detecting actual defects in the data. For ResNet-18 this indicates that on average 71.2 % of the various defects in the data were detected by the model. EfficientNetV2-S on the other hand classifies only 31.2 % of the defects correctly. Of further interest are the metrics combining recall and precision. A high value of PRC AUC indicates a good balance of recall and precision across various thresholds, and the F1 score measures the balance between the two metrics at the given threshold.

To address the question of the technology's suitability, it's essential to contextualize these results. In contrast to manual inspection, the results of the ResNets work in favor of the computer vision approach. According to the study of Buehlmann & Thomas (2002) on manual defect inspection, the false negative rate (FNR) was measured to be more than 43 %. In the case of ResNet-18, on average across all classes, around 29 % of the defects are incorrectly classified as non-defective, taking the FNR as $1 - \text{recall}$. Similarly, in the case of ResNet-50V2, the obtained value advocates for the use of the computer vision technology. However, this result cannot be transferred to the EfficientNet models. These models perform worse as compared to the manual inspection in the study, demonstrating FNRs of around 45 % (B0) and 69 % (S). However, compared to other computer vision methods investigated in research, there are some drawbacks of the examined technologies. While this study did not use accuracy as a performance metric, as it would be heavily skewed by true negatives in the case of multi-label classification, a general comparison is still feasible. The Mixed FCN approach by (He et al. 2019) reported precision values exceeding 98 % for the detection of six different defect types, and an overall classification accuracy of 99.14 %. However, in that study, the acquired image data were cropped and excess background was removed, resulting in at least one defect present in the majority of training samples. In the study by Gao, Qi, Mu & Chen (2021), per-class precision values of at least 92.5 % and per-class recall values of at least 95.5 % are achieved for seven different types of knot defects. The modified ResNet-18 from the study by Gao, Song, Wang, Liu, Mandelis & Qi (2021) reports per-class precision values of at least 97 %, per-class recall values of at least 94 %, and an overall accuracy of 99.92 %. These values are on average significantly higher than the averaged metrics acquired in this thesis. However, both of these studies represented a multi-class classification problem, where each input image consists of one specific knot defect. Hence, only the defect that is predicted as the most likely by the softmax function is detected. The approach investigated in this thesis has the potential to identify all defects present in the input and is therefore more specialized for an application in a production context.

Generally speaking, the best test result aligns with expectations about the performance of the models regarding the nature of the data used for the analysis. This is justified by the fact that the data set, consisting of 20,267 instances is small for computer vision problems. Upon partitioning into training, validation, and test sets, there are only 12,166 training instances remaining for

training purposes, of which about 10 % are hard negatives. The significance of the underlying data is emphasized when considering the results obtained through training on the use case data. Here, the performance is consistently poorer across all models compared to the results for the public data set. Particularly concerning precision and recall, and the combined metrics derived from them (PRC, F1), notable differences arise. Here, the EfficientNet models achieve values of 0 % for macro-averaged precision, recall, and F1 score using the pre-defined threshold for positive predictions of 0.5. The most probable reason for the poor performance of the models is the underlying data. The data set is approximately 13 times smaller than the public data set, and around half of the images are hard negatives. Due to the train-validation-test split, there are ultimately only 445 image data with defects available for training the models. Moreover, as with the public data set, the use case data exhibits significant imbalance between classes. The loss curves of the models indicate overfitting. The occurrence of overfitting is a typical sign of a small amount of data and relatively excessive complexity of the models.

Suitability for Wood Defect Detection

Based on the results of this investigation, the EfficientNetV2 variants are not suitable for defect detection in wood manufacturing. Despite the ResNet models outperforming human performance, the technology used comes with certain disadvantages that raise questions about its suitability for defect detection. In addition to insufficient performance, there are further disadvantages that are inherent to the nature of the computer vision task. On one hand, classification models only indicate the potential presence of one or more defects. However, in a production environment, the precise determination of the location of defects can also be crucial. This plays a significant role in defect assessment, potentially influencing the decision about rework or scrap. Within the scope of automation, the additional information obtained from detection could be used for machine or production control. Moreover, false positive results in image classification are not easily traceable and could lead to increased waste. Visualized bounding boxes in object detection can assist in avoiding false positives, i.e. when displayed on a monitor in production. Another important factor not covered by image classification algorithms is the severity of the detected defect. Images can only be examined for the presence of defects, while the size or extent of the defects remains unaccounted for. In summary, multi-label classification using the investigated CNN models and the available data is not an optimal solution for defect detection in wood processing production under the paradigm of ZDM.

8.1.2 Object Detection Models

Research question 1b focuses on the suitability of YOLOv7 and YOLOv8 object detectors for defect detection in wood manufacturing. For this purpose, the frameworks were trained on a variety of scales using an available dataset, and their performance was compared. The models were evaluated using various metrics, including mAP_{50}, mAP_{50-95}, as well as recall, and precision across classes. In terms of model speed, their frames per second (FPS) values for processing individual images were measured. To ensure comparability, all models were tested on the same hardware.

Comparison of Model Performance

In contrast to the image classification models, when comparing the cumulative object detection performance on the public dataset, it's noticeable that the variance in results for the performance metrics between the models is significantly lower. Only the base scaling of YOLOv7 stands out with lower values, particularly in the mAP metrics. YOLOv8-M achieved the best results in the categories of mAP50 and mAP50-95, with scores of 76.4 % and 45.4 %, respectively. Additionally, the model's recall was the highest with a score of 70.7 %. However, in terms of mean precision, the model is slightly outperformed by YOLOv8-S with 77.4 % vs. 75.3 %.

The rather similar performance of the models was not anticipated before analyzing the experiments. Particularly, considering that the smaller models, designed for applications on mobile devices, were expected to have a larger trade-off between detection performance and speed. One reason for the similar performance could be the small dataset on which the models were trained. This could result in the potential of the larger variants not being fully utilized. Additionally, relatively low learning rates were used, which might have prevented the models from finding better solutions due to insufficient convergence. For larger training datasets, it's expected that the performance of the larger models will show a more distinct difference from the smaller variants in terms of performance.

Regarding per-class precision results, YOLOv8-S emerged as the overall best alternative, being surpassed in only two classes. Interestingly, both YOLOv7 scalings outperform the values of the YOLOv8 frameworks in the blue stain class by a significant margin. This variance between the models could be caused by the architecture of the models, which determines how the models handle different features. Other potential reasons could include variations in hyperparameters or differences in the initialization between the architectures. The trade-off between high precision and high recall becomes evident when considering the per-class recall values. This observation is supported by the fact that YOLOv8-S is not the best-performing model for any of

the classes.

Contrary to expectations, the comparison of model speeds is not solely dependent on the complexity of the models. While the largest models, YOLOv7 and YOLOv8-L, are the slowest, the FPS values do not increase proportionally to the number of parameters for the other models. Surprisingly, YOLOv7-T emerges as the fastest model in the analysis, even though it has more FLOPs and parameters compared to YOLOv8-N. A similar pattern is observed when comparing YOLOv8-N and -S, with -S exhibiting superior FPS. Possible reasons for this discrepancy could be resource contention on the hardware or background processes affecting performance. Additionally, it's important to note that the models were evaluated on high-performance GPUs. In a real production implementation with less powerful hardware, significantly lower speeds should be anticipated.

Interpretation of Results

To determine how well the YOLOv7 and YOLOv8 models perform in the context of defect detection and localization in wood production, the results need to be examined with respect to this specific case.

The mAP_{50} value describes the average of the average precision across all classes at a defined IoU overlap of 50 %, thereby assessing how well the model performs in the detection and localization of defects in the image data. Apart from YOLOv7, all models exhibit values of at least 74 %, indicating robustness across various confidence thresholds. Additionally, in the comparison between test and validation mAP_{50}, no significant differences are observed, suggesting that excessive overfitting is not likely. However, the significant drop in the mAP_{50-95} values compared to the mAP_{50} values across all models indicates that the models suffer in terms of localization performance when higher IoU thresholds are applied. Reasons for this could be attributed to the high variability of the examined defects, which come in different forms and appearances. Additionally, the associated bounding box coordinates of the data set could influence these values. Accurately determining the extent of a defect can often be challenging when performed manually, leading to variations in the quality of bounding boxes during the labeling process.

Regarding per class precision, the values for quartzity and crack appear to be the lowest on average despite being not the least prevalent in the data set. In contrast to other classes like knots with distinct shapes, the absence of distinct features in the defects could lead to a higher number of false positives occuring for these classes. It also has to be noted that the average values for the blue stain class are distorted by the very high results from the YOLOv7 models.

Regarding per-class recall, the classes of blue stain and quartzity exhibit the lowest average val-

ues. The blue stain class is infrequently represented in the training dataset, which could result in the models not properly learning its features. Additionally, variations in appearance and ambiguous features could contribute to increased false negatives in these cases. On the other hand, marrow and live knot represent the classes with the highest average recall. In these cases, the models appear to have little difficulty in recognizing the defects. This could be attributed to distinct features or low variability in the appearance of the defects. Referring to the study of Buehlmann & Thomas (2002), the calculated average recall values indicate that all the presented models outperform the human operator.

Compared to the approach by (Li et al. 2021), the models examined in this thesis show lower mAP_{50} values in the object detection study. The study reports a mAP50 of 84.4 % for the proposed model. On the other hand, the model is significantly slower at around 15 FPS (NVIDIA V100) compared to the models presented here. However, it's important to note that the study only examines three types of defects, and the data consists of close-up shots of the defects.

In another study targeting three types of defects with a one-stage detector, Ding et al. (2020) achieved an mAP score of 96.1 % with a DenseNet-SSD algorithm, using close-up images of the respective defects as the data set. Compared to the results of this study, the performance is significantly higher. However, this comes at the expense of processing speed, which is reported to be around 18 FPS.

The Faster R-CNN approach from 2019 by (Urbonas et al. 2019) reports average precision and recall of 80.53 % and 80.65 %, respectively, which are also higher than the metrics results in this study. Specific mAP values are not provided and thus cannot be compared. However, it's important to note that the study only considers four defect classes in their investigation.

The modified YOLOv3 algorithms used by (Tu et al. 2021) achieved higher results with 92 % mAP and 86 % mAP on two different data sets, as compared to the study in this thesis. However, it is worth noting that the best result is based on a problem with only two defects, and in the other case, four defects. Additionally, no specific IoU threshold is provided based on which the mAP was calculated in their study, which complicates the comparison. On the other hand, the fundamental nature of the data is similar to that of this thesis regarding the content and appearance of the images.

Based on the results of the experiments and the comparison with other approaches, it is reasonable to assume that the models would perform better on more and higher-quality data or when reduced to fewer classes. Regarding the results of the models trained on the use case data, the importance of having sufficient and well-labeled data becomes even more evident. Significant discrepancies between test and validation mAP indicate overfitting. Due to the overall low amount of data with a substantial proportion of hard negatives, as well as the existing class

imbalance, the models are unable to effectively learn from the data. Additionally, the quality of the labeled bounding boxes likely plays a significant role, as indicated by the low values for mAP_{50-95}.

Suitability for Wood Defect Detection

To summarize the results it can be said that no model is superior at all tasks. For an application in the field of defect detection, the recall value holds the highest importance. This is because false negative detections can incur higher costs than false positive detections. As a result, YOLOv8-M might be the most suitable version out of the models analyzed. However, depending on the individual prioritization of certain defect types, a different model might be more suitable. All the investigated models process images very quickly, making them suitable for deployment in a fast-paced production environment, especially if the precise localization of defects is not extremely important. Furthermore, the use of object detectors also brings additional advantages, as described in section 8.1.1. However, it's important to note that the collected metrics were averaged without considering the imbalanced class distribution. In automated defect detection within wood product manufacturing, certain defects such as cracks might hold greater significance as they might not be immediately visible and could further develop during processing. Due to data imbalance, this results in a skewed view of the model's performance. To determine suitability for application in the wood manufacturing industry, classes might need to be examined separately or the metrics adjusted with weights as necessary. Setting a specific threshold value in order to minimize a certain error is crucial.

8.2 Effects of Offline Data Augmentation

Research question 2 focuses on evaluating the effectiveness of offline data augmentation for object detectors. Various augmentation operations were applied to the data, and the models trained on augmented data were tested against the baseline results. Augmentation was performed on both the public dataset and the use case dataset equally.

Based on reviewed studies, positive impacts on the performance and robustness of the models were expected. However, regarding the obtained results, this has only been partially confirmed.

In the case of the public dataset, improvements in mAP_{50} are observed in only three out of six cases. The same applies to the mean precision. The mean recall improves in four cases. However, it is noteworthy that the mAP_{50-95} has improved for five of the models. The better values across different IoU thresholds could be attributed to greater diversity introduced by the

additional modified data. The extra data with increased variability may enable the model to potentially generalize better to unknown instances. It is also interesting that YOLOv7 significantly benefits from offline augmentation compared to training on raw data. Possible reasons could be that the model, due to its complexity or architecture, requires more data compared to the other models under investigation to achieve good results. The augmented data set might have countered this issue by providing a larger set of data. Furthermore, it is possible that there were issues during training on the raw data, which could have caused the significant differences. Also noticeable is the poorer processing time during evaluation for all models. Besides possible external factors, reasons for this could include additional complexity introduced into the models as a result of training on augmented data.

In the case of the use case data, a similar pattern emerges. However, here the effects are sometimes more pronounced. The mAP_{50-95} could be improved in all cases through offline augmentation, which indicates greater robustness of the models across different IoU thresholds. The introduced variability from the additional and modified data could have led to this effect, making the model more robust against potentially different instances from the test set. Also noticeable is that in many cases, the average recall increases while the average precision decreases, and vice versa. This behavior could also be attributed to the introduced variation in the data.

Moreover, all of the models trained on augmented data seem to experience significant overfitting. This is indicated by the adverse behavior of validation loss compared to training loss. Reasons for this could lie in the chosen augmentation techniques. In this thesis, offline augmentation only involved a single geometric transformation. The other augmentation techniques were limited to color transformations and adding noise. This might have led to insufficient introduction of variability into the data, resulting in the observed overfitting. The lack of data diversity could have led the models to memorize the data rather than learning significant features of the defects. This could have also impacted their generalization ability, which might explain the partially poorer performance on the unchanged test data. In the case of the use case data, however, offline augmentation had a predominantly positive effect on the performance of the models. This could be attributed to the expansion of the very small data set with additional instances.

8.3 Effects of Transfer Learning

Research question 3 deals with the effects of transfer learning on object detection for defect detection in wood production. The question is divided into the effectiveness of transfer learning via pretraining on generic image data (3a) and via pretraining on other wood data. To answer research question 3a, the object detectors were initialized with weights pretrained on MS COCO,

then trained and evaluated on the public dataset. For research question 3b, the object detectors were initialized with the best weights from the augmentation study on public data, then trained and evaluated on the use case dataset. The results were compared to the baseline model results.

Pre-training Based on Generic Data

For the COCO-initialized models trained on the public data set, positive effects on training can be observed. These are reflected in faster improvements of the models and generally lower loss curves. Furthermore, the training duration was improved through transfer learning. This could be due to the fact that the models, through initialization with weights pre-trained on COCO, already possess knowledge about basic features and general patterns. In general, due to the diversity of the COCO data set, it is to be expected that some of the visual features are shared between the two different domains. Lower layers of the architectures could effectively reuse these features. As a result, this can help prevent initial difficulties during training compared to random initializations, resulting in faster convergence of the models. Another point in favor of the convergence speed could be the low-level statistics transferred by the initialization with pre-trained weights, in accordance to the findings of Neyshabur et al. (2020).

In terms of performance differences compared to training from scratch, a mixed picture emerges. The mAP_{50} slightly decreased for three of the YOLOv8 models, while YOLOv7 experienced a 6.5 percentage point drop. YOLOv7 also noticeably declined in mAP_{50-95}, whereas the other models showed improvements. The differences could be due to the model architectures and their sensitivity to pre-trained weights. Alternatively, differences in performance could also be attributed to hyperparameters. Regarding the values for mean precision and mean recall, a similar behavior was observed as with training on augmented data, where some models improve in recall at the expense of precision, and vice versa. Surprisingly, the YOLOv7-T model improved in all performance metrics compared to the larger YOLOv7 model. Reasons for this could be the lower complexity of the model and the resulting better fit to the available dataset and defined hyperparameters.

Pre-training Based on Wood Data

When training the use case data on the pre-trained weights from the augmented public dataset, significant improvements were anticipated. Advantages are reflected in the training process, with significantly improved convergence speeds in all cases, which could also be attributed to feature reuse. The performance comparison results show more improvements compared to the case with COCO weights. The values for mAP_{50} and mAP_{50-95} were enhanced in five out of six

cases each, and in several instances, by a few percentage points. The reason for this is believed to be the reuse of visually similar features from the same domain. This improvement helps the models avoid making false positive predictions. The mean precision was greatly improved in the cases of YOLOv8-N and -S, by 18.9 and 27.7 percentage points, respectively. At the same time, some models experienced a drop in mean recall. This behavior could be attributed to the specialization on the pretrained features, which may differ significantly in distribution from the features in the use case dataset. This divergence might lead to more cautious models, resulting in higher precision for certain classes but simultaneously reducing recall. Errors in the labeling process could also contribute to deteriorations, for instance, if defects were misclassified.

The results largely align with previous research. In the study by Norlander et al. (2015), pretraining on ImageNet improved the accuracy of a CNN for knot defect detection. The Mask R-CNN proposed by Hu et al. (2020) also improved by 4 percentage points through transfer learning with COCO. Similar findings regarding improvements in accuracy and convergence speed were reported by Gao, Qi, Mu & Chen (2021) and Gao, Song, Wang, Liu, Mandelis & Qi (2021) in relation to this thesis. It should be noted that in this study, transfer learning was conducted according to the official YOLOv7 tutorial (Wong 2022), and that no layer freezing was performed. Trial and error tests with frozen layers using YOLOv7 also revealed no improvements in performance.

8.4 Limitations

A central limitation of this work arises from the data set. Since not many large scale publicly available databases for images of wood defects exist, compromises had to be made, which is reflected in the size of the data set, the class distribution and the quality of the labels. Effective training of computer vision models requires a substantial amount of data. Due to the small data set size and imbalances within it, the best-possible performance of the examined methods could not be fully determined. Temporal limitations also led to the experiments not being conducted under optimal conditions. Extensive hyperparameter tuning and lengthy test runs were neglected due to limited hardware resources on the Idun cluster, associated with long waiting times for scheduled jobs. Instead, a trial-and-error approach was pursued, which might not have achieved the optimal solution. The obtained results could likely be improved with a sufficient data foundation and a structured optimization of the algorithms.

Additionally, due to delivery and scheduling difficulties, the use case could not be executed as originally planned in a real production environment. As a result, a last-minute decision was made to simulate the use case in a learning factory. However, this approach lacked the foun-

dation to gather a larger data set, prompting the need to replace it with self-acquired samples. Furthermore, deficiencies in tools and equipment within the learning factory led to delays and suboptimal solutions during the setup of the experiment. These issues are reflected in the collected use case data, which, due to its nature, is only of limited use for the underlying investigation. Another issue regarding use case data quality arises from the labeling process. While the data was labeled to the best of the available knowledge, some misjudgments and errors are likely to have occurred due to the lack of expert knowledge.

Chapter 9

Conclusion and Further Work

9.1 Conclusion

This thesis investigated how computer vision technologies can be effectively applied in the field of defect detection for wood production, under the framework of Industry 4.0 and ZDM concepts. The necessity of this research arose from the context that the competitiveness of companies in the wood processing industry is not sustainable due to economic and environmental factors when using traditional methods for quality assurance. For this purpose, four image classification models (ResNet-18, ResNet-50V2, EfficientNetV2-B0, EfficientNetV2-S) and six differently scaled object detection algorithms (YOLOv7-tiny, YOLOv7, YOLOv8-N, YOLOV8-S, YOLOv8-M, YOLOv8-L) were compared in terms of their performance and suitability. A publicly available image database covering various production-related wood defects was used as the data foundation. To examine the transferability to a real production setting, a prototype of a camera system was established within the scope of a use case. This system was employed to collect additional image data, which was subsequently used for additionally evaluating the models.

In the evaluation of the image classification models, significant differences in their performance on the used data sets were observed. On the publicly available data, ResNet-18 emerged as the overall best-performing model in terms of its ability to identify various defects. ResNet-50V2 also demonstrated acceptable performance, whereas the EfficientNetV2 models did not exhibit satisfactory performance within the scope of the study. On the use case data, none of the models performed well. Based on the investigation's results, it can be concluded that image classification is only partially suitable for defect detection in the production environment. Although the examined ResNet models were able to surpass the performance of manual inspection processes, there are disadvantages inherent to the nature of computer vision tasks that can lead to drawbacks in applying them for quality assurance in a production context, such as the lack of defect

localization and missing control mechanisms.

When examining the object detection algorithms on the publicly available dataset, less significant differences in performance between the models were observed. None of the examined models dominated in all performance categories. However, YOLOv8-M achieved the best results in the categories of mAP_{50}, mAP_{50-95}, and mean recall. YOLOv8-S exhibited the best performance in terms of precision. YOLOv7 represented the overall worst-performing model in the study. However, none of the examined models could demonstrate an acceptable performance on the use case data. Based on the results, the fundamental suitability of the models for defect detection in wood production can be determined. Furthermore, additional benefits of the computer vision task, such as providing information about the location, size, and severity of defects, can be utilized for intelligent production concepts. The output of bounding box information also provides an additional means of control.

An additional challenge in the domain of wood processing industry is the lack of data, primarily due to limited adoption of digital technologies and Industry 4.0 practices. Therefore, this thesis investigated whether effective improvements in model performance can be achieved with offline data augmentation or transfer learning methods, even when dealing with limited data sets.

To assess the effectiveness of offline augmentation techniques, both datasets were augmented using various methods. When compared to the baseline results, the outcomes were mixed for the publicly available dataset. The YOLOv8-L and YOLOv7 models were able to significantly enhance their detection performance. Other models exhibited improvements in certain metrics at the expense of deteriorations in other metrics. However, the models exhibited significant overfitting, which can be attributed to the lack of diversity in augmented data due to limited variations in the augmentation techniques. For the use case data, predominantly positive effects on the performance of the models were observed, with some instances of substantial improvements. In summary, it can be stated that, at least in scenarios with very limited data, offline data augmentation can be an effective tool for enhancing the performance. However, it is recommended to employ a more diverse range of augmentation techniques.

The effectiveness of transfer learning was investigated in two scenarios. When initializing models with pre-trained weights on generic image data, no clear improvement in performance was observed. However, significant improvements in convergence speed were noted. In the case of initializing with weights from models trained on wood images, training performance and convergence were also greatly enhanced. The performance development of the models under transfer learning was generally positive. In summary, it can be said that transfer learning is an effective tool to support the models in wood defect detection during training. However, due to the varied results, further investigations should be conducted for enhancing performance.

9.2 Recommendations for Further Work

The results obtained in this thesis have revealed issues due to the limited data set size and the existing class imbalance within the datasets. Therefore, it is recommended to generate more and better-distributed data and to train the tests on larger datasets. To achieve this, experts should be involved in the labeling process to ensure consistent high-quality labels and associated bounding box coordinates.

To further investigate the impacts of offline data augmentation in the field of wood defect detection, tests involving various geometric transformations should be conducted. Since the images in the existing data set are all in the same position, this approach could potentially enhance the models' robustness.

The effects of transfer learning using pre-trained models within the domain of wood defect detection should be validated with a significantly larger data set. This validation could determine whether the utilization of pre-trained models in real-world production scenarios can lead to significant improvements.

Furthermore, the models should be tested and optimized against different hyperparameter settings. This could be accomplished through heuristics or implemented optimization algorithms. The effects of training for more epochs should also be investigated.

Bibliography

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., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

Abdullah, N. D., Hashim, U. R., Ahmad, S. & Salahuddin, L. (2020), 'Analysis of texture features for wood defect classification', *Bulletin of Electrical Engineering and Informatics* **9**(1), 121–128.

Ahn, N. & Park, S. (2020), 'Heat transfer analysis of timber windows with different wood species and anatomical direction', *Energies* **13**(22), 6050.

Akachuku, A. & Abolarin, D. (1989), 'Variations in pith eccentricity and ring width in teak (tectona grandis l. f.)', *Trees* **3**(2).

Alexopoulos, K., Tsoukaladelis, T., Dimitrakopoulou, C., Nikolakis, N. & Eytan, A. (2023), 'An approach towards zero defect manufacturing by combining IIoT data with industrial social networking', *Procedia Computer Science* **217**, 403–412.

Arriaga, F., Wang, X., Íñiguez-González, G., Llana, D. F., Esteban, M. & Niemz, P. (2023), 'Mechanical properties of wood: A review', *Forests* **14**(6), 1202.

Bengio, Y., Simard, P. & Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE Transactions on Neural Networks* **5**(2), 157–166.

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020), 'Yolov4: Optimal speed and accuracy of object detection'.

Bond, B., Kline, D. & Araman, P. (1998), Characterization of defects in lumber using color, shape, and density information, *in* 'Proceedings of 1998 International Conference on Multisource-Multisensor Information Fusion, Las Vegas, NV, USA', pp. 581–587.

Broda, M. (2020), 'Natural compounds for wood protection against fungi—a review', *Molecules* **25**(15), 3538.

Buehlmann, U. & Thomas, R. E. (2002), 'Impact of human error on lumber yield in rough mills', *Robotics and Computer-Integrated Manufacturing* **18**(3-4), 197–203.

Bumgardner, M. & Buehlmann, U. (2022), 'A preliminary assessment of industry 4.0 and digitized manufacturing in the north american woodworking industry', *Forest Products Journal* **72**(1), 67–73.

Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V. I. & Kalinin, A. A. (2018), 'Albumentations: fast and flexible image augmentations', *Information* **11**(2), 125.

Butarbutar, T., Köhl, M. & Neupane, P. R. (2016), 'Harvested wood products and REDD: looking beyond the forest border', *Carbon Balance and Management* **11**(1).

Cao, J., Liang, H., Lin, X., Tu, W. & Zhang, Y. (2016), 'Potential of near-infrared spectroscopy to detect defects on the surface of solid wood boards', *BioResources* **12**(1).

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. & Zagoruyko, S. (2020), End-to-end object detection with transformers, *in* 'Computer Vision – ECCV 2020', Springer International Publishing, pp. 213–229.

Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P. & García-Gutiérrez, J. (2020), 'On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data', *Remote Sensing* **13**(1), 89.

Carratu, M., Gallo, V., Liguori, C., Pietrosanto, A., O'Nils, M. & Lundgren, J. (2021), A CNN-based approach to measure wood quality in timber bundle images, *in* '2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)', IEEE.

Chacon, M. I. & Alonso, G. R. (2006), Wood defects classification using a SOM/FFP approach with minimum dimension feature vector, *in* J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu & H. Yin, eds, 'Advances in Neural Networks - ISNN 2006', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1105–1110.

Chen, B., Wan, J., Shu, L., Li, P., Mukherjee, M. & Yin, B. (2018), 'Smart factory of industry 4.0: Key technologies, application case, and challenges', *IEEE Access* **6**, 6505–6519.

Chollet, F. (2020), 'Keras developer guides: Transfer learning & fine-tuning'. Accessed: 2023-06-08.
URL: *https://keras.io/guides/transfer_learning/*

Chollet, F. et al. (2015), 'Keras', `https://keras.io`.

Chun, T. H., Hashim, U. R., Ahmad, S., Salahuddin, L., Choon, N. H. & Kanchymalay, K. (2023), 'A review of the automated timber defect identification approach', *International Journal of Electrical and Computer Engineering (IJECE)* **13**(2), 2156.

Conners, R., Kline, D., Araman, P. & Drayer, T. (1997), 'Machine vision technology for the forest products industry', *Computer* **30**(7), 43–48.

Dai, J., Li, Y., He, K. & Sun, J. (2016), R-FCN: Object detection via region-based fully convolutional networks, *in* D. Lee, M. Sugiyama, U. Luxburg, I. Guyon & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 29, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper_files/paper/2016/file/577ef1154f3240ad5b9b413aa7346a1e-Paper.pdf*

Day, O. & Khoshgoftaar, T. M. (2017), 'A survey on heterogeneous transfer learning', *Journal of Big Data* **4**(1).

Demir, K. & Cicibaş, H. (2019), *The Next Industrial Revolution: Industry 5.0 and Discussions on Industry 4.0*, pp. 247–260.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009), ImageNet: A large-scale hierarchical image database, *in* '2009 IEEE Conference on Computer Vision and Pattern Recognition', IEEE.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'BERT: Pre-training of deep bidirectional transformers for language understanding'.

Ding, F., Zhuang, Z., Liu, Y., Jiang, D., Yan, X. & Wang, Z. (2020), 'Detecting defects on solid wood panels based on an improved SSD algorithm', *Sensors* **20**(18), 5315.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. (2020), 'An image is worth 16x16 words: Transformers for image recognition at scale'.

Dumoulin, V. & Visin, F. (2016), 'A guide to convolution arithmetic for deep learning'.

Dwyer, B., Nelson, J. & Solawetz, J. (2022), 'Roboflow (version 1.0) [software]', Available from `https://roboflow.com`. Computer Vision.

Eger, F., Coupek, D., Caputo, D., Colledani, M., Penalva, M., Ortiz, J. A., Freiberger, H. & Kollegger, G. (2018), 'Zero defect manufacturing strategies for reduction of scrap and inspection effort in multi-stage production systems', *Procedia CIRP* **67**, 368–373.

Eger, F., Tempel, P., Magnanini, M. C., Reiff, C., Colledani, M. & Verl, A. (2019), Part variation modeling in multi-stage production systems for zero-defect manufacturing, *in* '2019 IEEE International Conference on Industrial Technology (ICIT)', IEEE.

Ericsson, M., Johansson, D. & Stjern, D. (2021), 'AI-based quality control of wood surfaces with autonomous material handling', *Applied Sciences* **11**(21), 9965.

Estévez, P. A., Perez, C. A. & Goles, E. (2003), 'Genetic input selection to a neural classifier for defect classification of radiata pine boards', *Forest products journal* **53**(7/8), 87–94.

Estevez, P., Flores, R. & Perez, C. (2005), Color image segmentation using fuzzy min-max neural networks, *in* 'Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.', Vol. 5, pp. 3052–3057 vol. 5.

European Commission, Directorate-General for Research and Innovation, Breque, M., De Nul, L. & Petridis, A. (2021), *Industry 5.0 : towards a sustainable, human-centric and resilient European industry*, Publications Office of the European Union.

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. & Zisserman, A. (2009), 'The pascal visual object classes (VOC) challenge', *International Journal of Computer Vision* **88**(2), 303–338.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J. & Feichtenhofer, C. (2021), Multiscale vision transformers, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)', pp. 6824–6835.

Gao, D., Liu, Y., Huang, A., Ju, C., Yu, H. & Yang, Q. (2019), Privacy-preserving heterogeneous federated transfer learning, *in* '2019 IEEE International Conference on Big Data (Big Data)', IEEE.

Gao, M., Qi, D., Mu, H. & Chen, J. (2021), 'A transfer residual neural network based on ResNet-34 for detection of wood knot defects', *Forests* **12**(2), 212.

Gao, M., Song, P., Wang, F., Liu, J., Mandelis, A. & Qi, D. (2021), 'A novel deep convolutional neural network based on ResNet-18 and transfer learning for detection of wood knot defects', *Journal of Sensors* **2021**, 1–16.

Gao, M., Wang, F., Liu, J., Song, P., Chen, J., Yang, H., Mu, H., Qi, D., Chen, M., Wang, Y. & Yue, H. (2022), 'Estimation of the convolutional neural network with attention mechanism and transfer learning on wood knot defect classification', *Journal of Applied Physics* **131**(23), 233101.

Gauder, D., Bott, A., Gölz, J. & Lanza, G. (2023), 'Simulation uncertainty determination of single flank rolling tests using monte carlo simulation and skin model shapes for zero defect manufacturing of micro gears', *Computers in Industry* **146**, 103854.

Gazo, R., Vanek, J., Abdul_Massih, M. & Benes, B. (2020), 'A fast pith detection for computed tomography scanned hardwood logs', *Computers and Electronics in Agriculture* **170**, 105107.

Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. (2021), 'YOLOX: Exceeding yolo series in 2021'.

Geng, A., Yang, H., Chen, J. & Hong, Y. (2017), 'Review of carbon storage function of harvested wood products and the potential of wood substitution in greenhouse gas mitigation', *Forest Policy and Economics* **85**, 192–200.

Ghobakhloo, M. (2020), 'Industry 4.0, digitization, and opportunities for sustainability', *Journal of Cleaner Production* **252**, 119869.

Girshick, R. (2015), Fast R-CNN, *in* '2015 IEEE International Conference on Computer Vision (ICCV)', IEEE.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* Y. W. Teh & M. Titterington, eds, 'Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics', Vol. 9 of *Proceedings of Machine Learning Research*, PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256.
URL: *https://proceedings.mlr.press/v9/glorot10a.html*

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), 'Generative adversarial nets', *Advances in Neural Information Processing Systems* **27**.
URL: *https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf*

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2020), 'Generative adversarial networks', *Commun. ACM* **63**(11), 139–144.
URL: *https://doi.org/10.1145/3422622*

Gu, I. Y.-H., Andersson, H. & Vicen, R. (2009), 'Wood defect classification based on image analysis and support vector machines', *Wood Science and Technology* **44**(4), 693–704.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. & Chen, T. (2018), 'Recent advances in convolutional neural networks', *Pattern Recognition* **77**, 354–377.

Habite, T., Abdeljaber, O. & Olsson, A. (2021), 'Automatic detection of annual rings and pith location along norway spruce timber boards using conditional adversarial networks', *Wood Science and Technology* **55**(2), 461–488.

Habite, T., Abdeljaber, O. & Olsson, A. (2022), 'Determination of pith location along norway spruce timber boards using one dimensional convolutional neural networks trained on virtual timber boards, woo', *Construction and Building Materials* **329**, 127129.

Hashim, U. R., Hashim, S. Z. & Muda, A. K. (2015), 'Automated vision inspection of timber surface defect: A review', *Jurnal Teknologi* **77**(20).

He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R. (2020), Momentum contrast for unsupervised visual representation learning, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)'.

He, K., Gkioxari, G., Dollar, P. & Girshick, R. (2017), Mask R-CNN, *in* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)'.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *in* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)'.

He, K., Zhang, X., Ren, S. & Sun, J. (2016*a*), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

He, K., Zhang, X., Ren, S. & Sun, J. (2016*b*), Identity mappings in deep residual networks, *in* 'Computer Vision – ECCV 2016', Springer International Publishing, pp. 630–645.

He, T., Liu, Y., Xu, C., Zhou, X., Hu, Z. & Fan, J. (2019), 'A fully convolutional neural network for wood defect location and identification', *IEEE Access* **7**, 123453–123462.

Heräjärvi, H., Kunttu, J., Hurmekoski, E. & Hujala, T. (2019), 'Outlook for modified wood use and regulations in circular economy', *Holzforschung* **74**(4), 334–343.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors'.

Hofmann, E. & Rüsch, M. (2017), 'Industry 4.0 and the current status as well as future prospects on logistics', *Computers in Industry* **89**, 23–34.

Hu, K., Wang, B., Shen, Y., Guan, J. & Cai, Y. (2020), 'Defect identification method for poplar veneer based on progressive growing generated adversarial network and mask R-CNN model', *BioResources* **15**(2), 3041–3052.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. & Murphy, K. (2017), Speed/accuracy trade-offs for modern convolutional object detectors, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Hwang, S.-W., Lee, T., Kim, H., Chung, H., Choi, J. G. & Yeo, H. (2021), 'Classification of wood knots using artificial neural networks with texture and local feature-based image descriptors', *Holzforschung* **76**(1), 1–13.

Hwang, S.-W. & Sugiyama, J. (2021), 'Computer vision-based wood identification and its expansion and contribution potentials in wood science: A review', *Plant Methods* **17**(1).

Iakubovskii, P. (2018), 'Classification models zoo - keras (and tensorflow keras)', `https://github.com/qubvel/classification_models`.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* F. Bach & D. Blei, eds, 'Proceedings of the 32nd International Conference on Machine Learning', Vol. 37 of *Proceedings of Machine Learning Research*, PMLR, Lille, France, pp. 448–456.
URL: *https://proceedings.mlr.press/v37/ioffe15.html*

Issa, A., Hatiboglu, B., Bildstein, A. & Bauernhansl, T. (2018), 'Industrie 4.0 roadmap: Framework for digital transformation based on the concepts of capability maturity and alignment', *Procedia CIRP* **72**, 973–978.

Ji, X., Guo, H. & Hu, M. (2019), Features extraction and classification of wood defect based on hu invariant moment and wavelet moment and BP neural network, *in* 'Proceedings of the 12th International Symposium on Visual Information Communication and Interaction', ACM.

Jocher, G., Chaurasia, A. & Qiu, J. (2023), 'YOLO by Ultralytics'.
URL: *https://github.com/ultralytics/ultralytics*

Kagermann, H., Lukas, W.-D. & Wahlster, W. (2011), 'Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution'.
URL: *https://www.dfki.de/fileadmin/user_upload/DFKI/Medien/News_Media/Presse/Presse-Highlights/vdinach2011a13-ind4.0-Internet-Dinge.pdf*

Kagermann, H. & Wahlster, W. (2022), 'Ten years of industrie 4.0', *Sci* **4**(3), 26.

Kagermann, H., Wahlster, W., Helbig, J. et al. (2013), 'Recommendations for implementing the strategic initiative industrie 4.0: Final report of the industrie 4.0 working group', *Forschungsunion: Berlin, Germany* .

Kamal, K., Qayyum, R., Mathavan, S. & Zafar, T. (2017), 'Wood defects classification using laws texture energy measures and supervised learning approach', *Advanced Engineering Informatics* **34**, 125–135.

Kayo, C., Tsunetsugu, Y. & Tonosaki, M. (2015), 'Climate change mitigation effect of harvested wood products in regions of japan', *Carbon Balance and Management* **10**(1).

Ke, Z.-N., Zhao, Q.-J., Huang, C.-H., Ai, P. & Yi, J.-G. (2016), Detection of wood surface defects based on particle swarm-genetic hybrid algorithm, *in* '2016 International Conference on Audio, Language and Image Processing (ICALIP)', IEEE.

Kline, D., Surak, C. & Araman, P. A. (2003), 'Automated hardwood lumber grading utilizing a multiple sensor machine vision technology', *Computers and Electronics in Agriculture* **41**(1-3), 139–155.

Kodytek, P. & Bodzas, A. (2021), 'Supporting tools for managing and labeling raw wood defect images.'.

Kodytek, P., Bodzas, A. & Bilik, P. (2022), 'A large-scale image dataset of wood surface defects for automated vision-based quality control processes', *F1000Research* **10**, 581.

Kodytek Pavel, Bodzas Alexandra & Bilik Petr (2021), 'Supporting data for deep learning and machine vision based approaches for automated wood defect detection and quality control.'.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S. & Houlsby, N. (2020), Big transfer (BiT): General visual representation learning, *in* 'Computer Vision – ECCV 2020', Springer International Publishing, pp. 491–507.

Kollmann, F. F. P. & Côté, W. A. (1968), *Principles of Wood Science and Technology*, Springer Berlin Heidelberg.

Koman, S., Feher, S., Abraham, J., Taschner, R. et al. (2013), 'Effect of knots on the bending strength and the modulus of elasticity of wood', *Wood Research* **58**(4), 617–626.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1', NIPS'12, Curran Associates Inc., Red Hook, NY, USA, p. 1097–1105.

Krogh, A. & Hertz, J. (1991), A simple weight decay can improve generalization, *in* J. Moody, S. Hanson & R. Lippmann, eds, 'Advances in Neural Information Processing Systems', Vol. 4, Morgan-Kaufmann.
URL: *https://proceedings.neurips.cc/paper_files/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf*

Kromoser, B., Reichenbach, S., Hellmayr, R., Myna, R. & Wimmer, R. (2022), 'Circular economy in wood construction – additive manufacturing of fully recyclable walls made from renewables: Proof of concept and preliminary data', *Construction and Building Materials* **344**, 128219.

Kryl, M., Danys, L., Jaros, R., Martinek, R., Kodytek, P. & Bilik, P. (2020), 'Wood recognition and quality imaging inspection systems', *Journal of Sensors* **2020**, 1–19.

Landscheidt, S. & Kans, M. (2016), Automation practices in wood product industries: Lessons learned, current practices and future perspectives, *in* 'The 7th Swedish Production Symposium SPS, 25-27 October, 2016, Lund, Sweden', Lund University.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *Nature* **521**(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989), 'Backpropagation applied to handwritten zip code recognition', *Neural Computation* **1**(4), 541–551.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Leskinen, P., Cardellini, G., González-García, S., Hurmekoski, E., Sathre, R., Seppälä, J., Smyth, C., Stern, T. & Verkerk, P. J. (2018), Substitution effects of wood-based products in climate change mitigation, Technical report.

Li, D., Xie, W., Wang, B., Zhong, W. & Wang, H. (2021), 'Data augmentation and layered deformable mask r-CNN-based detection of wood defects', *IEEE Access* **9**, 108162–108174.

Li, X., Chen, S., Hu, X. & Yang, J. (2019), Understanding the disharmony between dropout and batch normalization by variance shift, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)'.

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J. & Yang, J. (2020), Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan & H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 21002–21012.
URL: *https://proceedings.neurips.cc/paper_files/paper/2020/file/f0bda020d2470f2e74990a07a607ebd9-Paper.pdf*

Lin, H.-I. & Sanjaya, S. D. (2021), Wood polish classification for automated quality inspection based on AI vision, *in* '2021 21st International Conference on Control, Automation and Systems (ICCAS)', IEEE.

Lin, T.-Y. & Dollar, P. (2016), 'Coco api'. Accessed: 2023-07-05.
URL: *https://github.com/cocodataset/cocoapi*

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B. & Belongie, S. (2017), Feature pyramid networks for object detection, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollar, P. (2017), Focal loss for dense object detection, *in* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)'.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014), Microsoft COCO: Common objects in context, *in* 'Computer Vision – ECCV 2014', Springer International Publishing, pp. 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A. C. (2015), 'Ssd: Single shot multibox detector', pp. 21–37.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. & Guo, B. (2021), Swin transformer: Hierarchical vision transformer using shifted windows, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 10012–10022.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. & Xie, S. (2022), A convnet for the 2020s, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 11976–11986.

Long, J., Shelhamer, E. & Darrell, T. (2015), Fully convolutional networks for semantic segmentation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Loukadakis, M., Cano, J. & O'Boyle, M. (2018), Accelerating deep neural networks on low power heterogeneous architectures, *in* '11th International Workshop on Programmability and Architectures for Heterogeneous Multicores (MULTIPROG-2018)'. 11th International Workshop on Programmability and Architectures for Heterogeneous Multicores (MULTIPROG-2018), MULTIPROG-2018 ; Conference date: 24-01-2018.
URL: *http://research.ac.upc.edu/multiprog/*

Misra, D. (2019), 'Mish: A self regularized non-monotonic activation function'.

Mittal, S., Khan, M. A., Romero, D. & Wuest, T. (2018), 'A critical review of smart manufacturing &amp industry 4.0 maturity models: Implications for small and medium-sized enterprises (SMEs)', *Journal of Manufacturing Systems* **49**, 194–214.

Molinaro, M. & Orzes, G. (2022), 'From forest to finished products: The contribution of industry 4.0 technologies to the wood sector', *Computers in Industry* **138**, 103637.

Muñoz, G. R. & Gete, A. R. (2012), 'Prediction of bending strength in oak beams on the basis of elasticity, density, and wood defects', *Journal of Materials in Civil Engineering* **24**(6), 629–634.

Müller, J. M. (2019), 'Business model innovation in small- and medium-sized enterprises', *Journal of Manufacturing Technology Management* **30**(8), 1127–1142.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Neyshabur, B., Sedghi, H. & Zhang, C. (2020), What is being transferred in transfer learning?, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan & H. Lin, eds, 'Advances in Neural Information Processing Systems', Vol. 33, Curran Associates, Inc., pp. 512–523.
URL: *https://proceedings.neurips.cc/paper_files/paper/2020/file/0607f4c705595b911a4f3e7a127b44e0-Paper.pdf*

Norlander, R., Grahn, J. & Maki, A. (2015), Wooden knot detection using ConvNet transfer learning, *in* 'Image Analysis', Springer International Publishing, pp. 263–274.

Oksuz, K., Cam, B. C., Kalkan, S. & Akbas, E. (2021), 'Imbalance problems in object detection: A review', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**(10), 3388–3415.

Pan, S. J. & Yang, Q. (2010), 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering* **22**(10), 1345–1359.

Pascanu, R., Mikolov, T. & Bengio, Y. (2013), On the difficulty of training recurrent neural networks, *in* S. Dasgupta & D. McAllester, eds, 'Proceedings of the 30th International Conference on Machine Learning', Vol. 28 of *Proceedings of Machine Learning Research*, PMLR, Atlanta, Georgia, USA, pp. 1310–1318.
URL: *https://proceedings.mlr.press/v28/pascanu13.html*

Pech, M. & Vrchota, J. (2020), 'Classification of small- and medium-sized enterprises based on the level of industry 4.0 implementation', *Applied Sciences* **10**(15), 5150.

Powell, D., Magnanini, M. C., Colledani, M. & Myklebust, O. (2022), 'Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions', *Computers in Industry* **136**, 103596.

Psarommatis, F., May, G., Dreyfus, P.-A. & Kiritsis, D. (2019), 'Zero defect manufacturing: state-of-the-art review, shortcomings and future directions in research', *International Journal of Production Research* **58**(1), 1–17.

Qayyum, R., Kamal, K., Zafar, T. & Mathavan, S. (2016), Wood defects classification using GLCM based features and PSO trained neural network, *in* '2016 22nd International Conference on Automation and Computing (ICAC)', IEEE.

Raabe, H., Myklebust, O. & Eleftheriadis, R. (2018), Vision based quality control and maintenance in high volume production by use of zero defect strategies, *in* 'Lecture Notes in Electrical Engineering', Springer Singapore, pp. 405–412.

Raghu, M., Zhang, C., Kleinberg, J. & Bengio, S. (2019), Transfusion: Understanding transfer learning for medical imaging, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 32, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper_files/paper/2019/file/eb1e78328c46506b46a4ac4a1e378b91-Paper.pdf*

Rahiddin, R. N. N., Hashim, U. R., Ismail, N. H., Salahuddin, L., Choon, N. H. & Zabri, S. N. (2020), 'Classification of wood defect images using local binary pattern variants', *International Journal of Advances in Intelligent Informatics* **6**(1), 36.

Ramachandran, P., Zoph, B. & Le, Q. V. (2017), 'Searching for activation functions'.

Ramage, M. H., Burridge, H., Busse-Wicher, M., Fereday, G., Reynolds, T., Shah, D. U., Wu, G., Yu, L., Fleming, P., Densley-Tingley, D., Allwood, J., Dupree, P., Linden, P. & Scherman, O. (2017), 'The wood from the trees: The use of timber in construction', *Renewable and Sustainable Energy Reviews* **68**, 333–359.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Redmon, J. & Farhadi, A. (2018), 'Yolov3: An incremental improvement'.

Reis, D., Kupec, J., Hong, J. & Daoudi, A. (2023), 'Real-time flying object detection with yolov8'.

Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster R-CNN: Towards real-time object detection with region proposal networks, *in* C. Cortes, N. Lawrence, D. Lee, M. Sugiyama & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 28, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf*

Ronneberger, O., Fischer, P. & Brox, T. (2015), U-net: Convolutional networks for biomedical image segmentation, *in* 'Lecture Notes in Computer Science', Springer International Publishing, pp. 234–241.

Ross, R. J. (2015), Nondestructive evaluation of wood: Second edition, Technical report.

Ross, R. J., Brashaw, B. K. & Pellerin, R. F. (1998), 'Nondestructive evaluation of wood', *Forest Products Journal* **48**(1), 14–19. Copyright - Copyright Forest Products Society Jan 1998; Last updated - 2023-02-23; CODEN - FPJOAB; SubjectsTermNotLitGenreText - United States–US.
URL: *https://www.proquest.com/scholarly-journals/nondestructive-evaluation-wood/docview/214629527/se-2*

Ruder, S., Peters, M. E., Swayamdipta, S. & Wolf, T. (2019), Transfer learning in natural language processing, *in* 'Proceedings of the 2019 Conference of the North', Association for Computational Linguistics.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *Nature* **323**(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015), 'ImageNet large scale visual recognition challenge', *International Journal of Computer Vision* **115**(3), 211–252.

Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P. & Harnisch, M. (2015), 'Industry 4.0: The future of productivity and growth in manufacturing industries', *Boston consulting group* **9**(1), 54–89.

Ruz, G. A., Estévez, P. A. & Ramírez, P. A. (2009), 'Automated visual inspection system for wood defect classification using computational intelligence techniques', *International Journal of Systems Science* **40**(2), 163–172.

Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. (2018), How does batch normalization help optimization?, *in* S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett, eds, 'Advances in Neural Information Processing Systems', Vol. 31, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf*

Seppälä, J., Heinonen, T., Pukkala, T., Kilpeläinen, A., Mattila, T., Myllyviita, T., Asikainen, A. & Peltola, H. (2019), 'Effect of increased wood harvesting and utilization on required greenhouse gas displacement factors of wood-based products and fuels', *Journal of Environmental Management* **247**, 580–587.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. & LeCun, Y. (2013), 'Overfeat: Integrated recognition, localization and detection using convolutional networks'.

Sharma, S., Xing, C., Liu, Y. & Kang, Y. (2019), Secure and efficient federated transfer learning, *in* '2019 IEEE International Conference on Big Data (Big Data)', IEEE.

Shi, J., Li, Z., Zhu, T., Wang, D. & Ni, C. (2020), 'Defect detection of industry wood veneer based on NAS and multi-channel mask r-CNN', *Sensors* **20**(16), 4398.

Shorten, C. & Khoshgoftaar, T. M. (2019), 'A survey on image data augmentation for deep learning', *Journal of Big Data* **6**(1).

Silvén, O., Niskanen, M. & Kauppinen, H. (2003), 'Wood inspection with non-supervised clustering', *Machine Vision and Applications* **13**(5-6), 275–285.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition'.

Smyth, C., Rampley, G., Lemprière, T. C., Schwab, O. & Kurz, W. A. (2016), 'Estimating product and energy substitution benefits in national-scale mitigation analyses for canada', *GCB Bioenergy* **9**(6), 1071–1084.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research* **15**(56), 1929–1958.
URL: *http://jmlr.org/papers/v15/srivastava14a.html*

Stentoft, J., Wickstrøm, K. A., Philipsen, K. & Haug, A. (2020), 'Drivers and barriers for industry 4.0 readiness and practice: empirical evidence from small and medium-sized manufacturers', *Production Planning &amp Control* **32**(10), 811–828.

Sun, P. (2022), 'Wood quality defect detection based on deep learning and multicriteria framework', *Mathematical Problems in Engineering* **2022**, 1–9.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. & Liu, C. (2018), A survey on deep transfer learning, *in* 'Artificial Neural Networks and Machine Learning – ICANN 2018', Springer International Publishing, pp. 270–279.

Tan, M. & Le, Q. (2019), EfficientNet: Rethinking model scaling for convolutional neural networks, *in* K. Chaudhuri & R. Salakhutdinov, eds, 'Proceedings of the 36th International Conference on Machine Learning', Vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 6105–6114.
URL: *https://proceedings.mlr.press/v97/tan19a.html*

Tan, M. & Le, Q. (2021), Efficientnetv2: Smaller models and faster training, *in* M. Meila & T. Zhang, eds, 'Proceedings of the 38th International Conference on Machine Learning', Vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 10096–10106.
URL: *https://proceedings.mlr.press/v139/tan21a.html*

Terven, J. & Cordova-Esparza, D. (2023), 'A comprehensive review of yolo: From yolov1 and beyond'.

Tian, Z., Shen, C., Chen, H. & He, T. (2019), Fcos: Fully convolutional one-stage object detection, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)'.

Tu, Y., Ling, Z., Guo, S. & Wen, H. (2021), 'An Accurate and Real-Time Surface Defects Detection Method for Sawn Lumber', *IEEE Transactions on Instrumentation and Measurement* **70**, 1–11.

Urbonas, A., Raudonis, V., Maskeliūnas, R. & Damaševičius, R. (2019), 'Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning', *Applied Sciences* **9**(22), 4898.

Uzen, H., Turkoglu, M. & Hanbay, D. (2021), 'Texture defect classification with multiple pooling and filter ensemble based on deep neural network', *Expert Systems with Applications* **175**, 114838.

Uzunovic, A., Byrne, T., Gignac, M. & Yang, D.-Q. (2008), 'Wood discolourations and their prevention', *FP Innovations. Canada* .

Vaidya, S., Ambad, P. & Bhosle, S. (2018), 'Industry 4.0 – a glimpse', *Procedia Manufacturing* **20**, 233–238.

Vaillant, R. (1994), 'Original approach for the localisation of objects in images', *IEE Proceedings - Vision, Image, and Signal Processing* **141**(4), 245.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. (2023), Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 7464–7475.

Wang, C.-Y., Liao, H.-Y. M. & Yeh, I.-H. (2022), 'Designing network design strategies through gradient path analysis'.

Wang, C.-Y., Yeh, I.-H. & Liao, H.-Y. M. (2021), 'You only learn one representation: Unified network for multiple tasks'.

Wang, K. I.-K., Zhou, X., Liang, W., Yan, Z. & She, J. (2022), 'Federated transfer learning based cross-domain prediction for smart manufacturing', *IEEE Transactions on Industrial Informatics* **18**(6), 4088–4096.

Wang, S. & Su, Z. (2019), 'Metamorphic testing for object detection systems'.

Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., Zhao, Y. & Yan, S. (2016), 'HCP: A flexible CNN framework for multi-label image classification', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38**(9), 1901–1907.

Weiss, K., Khoshgoftaar, T. M. & Wang, D. (2016), 'A survey of transfer learning', *Journal of Big Data* **3**(1).

Wong, K.-Y. (2022), 'Official yolov7', `https://github.com/WongKinYiu/yolov7`.

Yang, Y., Zhou, X., Liu, Y., Hu, Z. & Ding, F. (2020), 'Wood defect detection based on depth extreme learning machine', *Applied Sciences* **10**(21), 7488.

Yu, H., Liang, Y., Liang, H. & Zhang, Y. (2019), 'Recognition of wood surface defects with near infrared spectroscopy and machine vision', *Journal of Forestry Research* **30**(6), 2379–2386.

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, *in* 'Computer Vision – ECCV 2014', Springer International Publishing, pp. 818–833.

Zhang, Q. & Yang, Y.-B. (2021), ResT: An efficient transformer for visual recognition, *in* M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang & J. W. Vaughan, eds, 'Advances in Neural Information Processing Systems', Vol. 34, Curran Associates, Inc., pp. 15475–15485.
URL: *https://proceedings.neurips.cc/paper_files/paper/2021/file/82c2559140b95ccda9c6ca4a8b981f1e-Paper.pdf*

Zhang, S., Chi, C., Yao, Y., Lei, Z. & Li, S. Z. (2020), Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)'.

Zhang, Y., Xu, C., Li, C., Yu, H. & Cao, J. (2015), 'Wood defect detection method with PCA feature fusion and compressed sensing', *Journal of Forestry Research* **26**(3), 745–751.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R. & Ren, D. (2020), 'Distance-IoU loss: Faster and better learning for bounding box regression', *Proceedings of the AAAI Conference on Artificial Intelligence* **34**(07), 12993–13000.

Zhou, X., Wang, D. & Krähenbühl, P. (2019), 'Objects as points'.

Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. (2017), Unpaired image-to-image translation using cycle-consistent adversarial networks, *in* 'Proceedings of the IEEE International Conference on Computer Vision (ICCV)'.

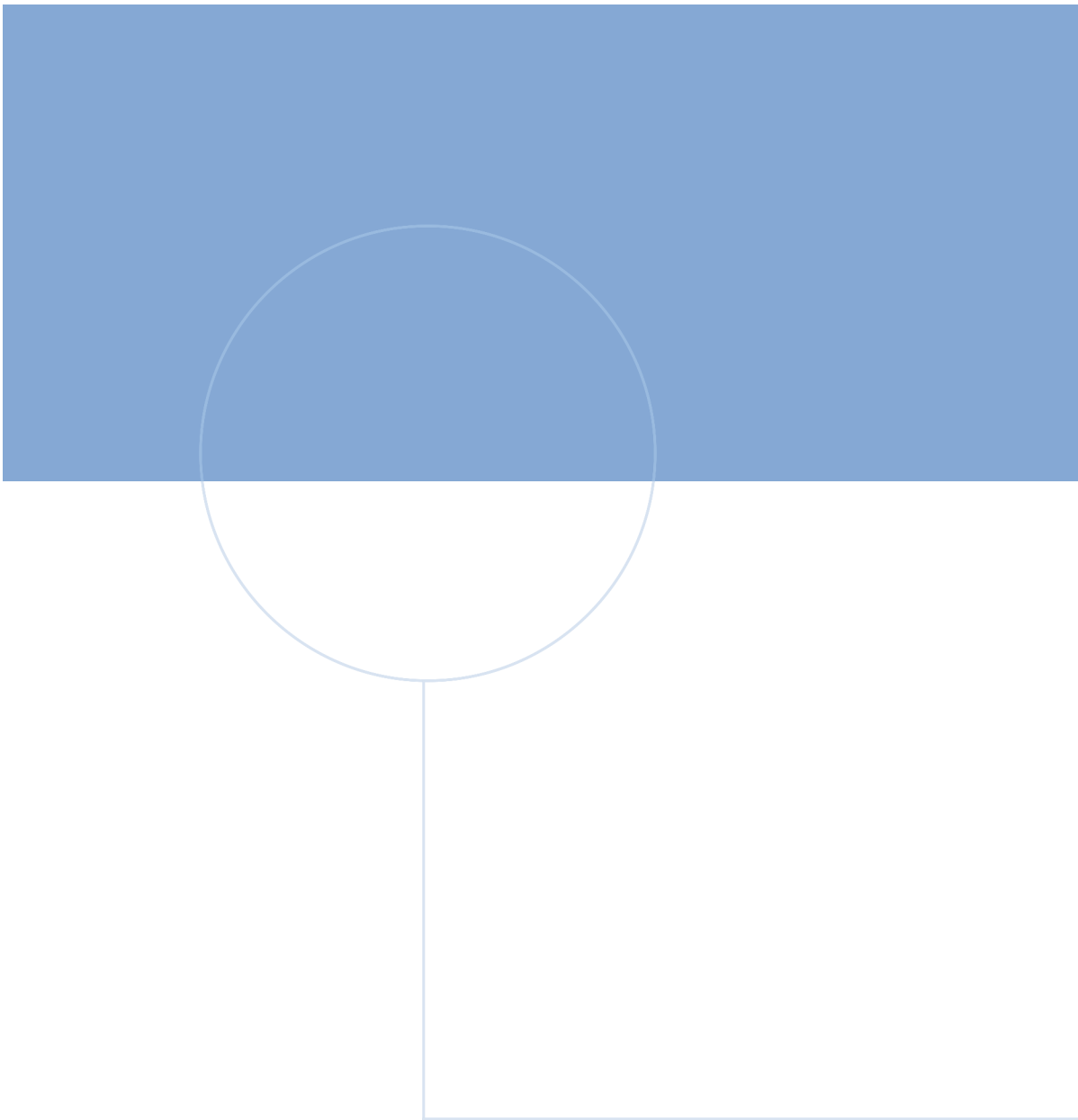
Zhu, X., Hu, H., Lin, S. & Dai, J. (2019), Deformable convnets v2: More deformable, better results, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)'.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. & He, Q. (2021), 'A comprehensive survey on transfer learning', *Proceedings of the IEEE* **109**(1), 43–76.

Zielińska, M. & Rucka, M. (2021), 'Non-destructive testing of wooden elements', *IOP Conference Series: Materials Science and Engineering* **1203**(3), 032058.

List of Figures

List of Tables



NTNU

Norwegian University of
Science and Technology