Diploma Thesis

Automatic evaluation of experiments based on machine learning

carried out for the purpose of obtaining the degree of
Diplom-Ingenieur (Dipl.-Ing. or DI),

submitted at TU Wien
Faculty of Mechanical and Industrial Engineering

by

Lorenz Michael Pfanner, BSc

Mat.Nr.: 01614605

under the supervision of
Univ.Ass. **Clemens David Fricke**, MSc
Associate Prof. Dipl.-Ing. Dr.techn. **Heinz Pettermann**

Institute of Lightweight Design and Structural Biomechanics (E317)

Vienna, November 2023

Signature

This work has been supported by 4a engineering GmbH.

I confirm that the printing of this thesis requires the approval of the examination board.

Affidavit

I declare in lieu of oath, that I wrote this thesis and carried out the associated research myself, using only the literature cited in this volume. If text passages from sources are used literally, they are marked as such.

I confirm that this work is original and has not been submitted for examination elsewhere, nor is it currently under consideration for a thesis elsewhere. I acknowledge that the submitted work will be checked electronically-technically using suitable and state-of-the-art means (plagiarism detection software). On the one hand, this ensures that the submitted work was prepared according to the high-quality standards within the applicable rules to ensure good scientific practice "Code of Conduct" at the TU Wien. On the other hand, a comparison with other student theses avoids violations of my personal copyright.

Vienna, November 2023

Signature

Acknowledgements

I would like to express my deepest gratitude to Associate Prof. Dipl.-Ing. Dr.techn Heinz Pettermann and Univ.Ass. Clemens David Fricke, MSc. for their excellent support during all stages of the thesis in the past months. Their generous allocation of time has been greatly appreciated and instrumental in contributing to the successful outcome of this work.

Additionally, I would like to sincerely thank Dr. Martin Schwab, Bernhard Jilka, and the entire 4a engineering GmbH, who made this work possible for me. Working on this topic in a practical environment and expanding my skills into new areas was fascinating.

I also want to thank my fellow students, friends, and those closest to my heart, who have contributed significantly to the success of my studies over the past years and are an important anchor point in my life. Not only have they made my time as a student exciting, but they have also strongly supported my personal growth.

Finally, I would like to express my heartfelt gratitude to my family for their unconditional support and encouragement. Their love and faith in me are a constant source of strength.

Contents

Abstract I

Kurzfassung II

Acronyms IV

List of Figures V

List of Tables VII

1 Introduction 1
- 1.1 Problem Statement . . . 2
- 1.2 Methodology and Structure . . . 3

2 Introduction to Material Testing 6
- 2.1 Force-Displacement and Stress-Strain Curve . . . 6
- 2.2 Experimental Setup and Testing Procedure . . . 9

3 State-of-the-Art: Computer Vision and Machine Learning 13
- 3.1 Image Processing . . . 15
  - 3.1.1 Point Operations . . . 15
  - 3.1.2 Digital Image Filtering - Convolution . . . 16
- 3.2 Texture Features - Haralick Features . . . 18
- 3.3 Segmentation vs. Classification . . . 21
- 3.4 Changepoint Detection . . . 21
- 3.5 Deep Learning . . . 22
  - 3.5.1 Neuron and Fully Connected Neural Network . . . 23
  - 3.5.2 Activation Functions . . . 24
  - 3.5.3 Loss Functions . . . 26
- 3.6 Convolutional Neural Networks . . . 27
  - 3.6.1 Convolution and Pooling Layers . . . 27
  - 3.6.2 Network Architectures . . . 29

3.7 Training of Neural Networks . . . 34
3.7.1 Optimization . . . 35
3.7.2 Transfer learning . . . 37
3.7.3 Overfitting and Dropout . . . 38

4 Literature Analysis 39
4.1 Results from the Literature Analysis . . . 39

5 Exploratory Data Analysis 42
5.1 Introduction to the provided data . . . 42
5.2 Results of the Data Analysis . . . 44

6 Used Approaches 45

7 Implementation 47
7.1 Pipeline . . . 47
7.1.1 Image Series . . . 48
7.1.2 Pre-Processing . . . 48
7.1.3 Model Training . . . 49
7.1.4 Processing/Predicting . . . 50
7.1.5 Detection . . . 50
7.1.6 Results . . . 51
7.2 Approach 1: Simple Image Processing . . . 51
7.2.1 Implementation of the Simple Image Processing . . . 51
7.3 Approach 2: Haralick Features and Neural Network . . . 53
7.3.1 Implementation of the Haralick Features and Neural Network . . . 53
7.4 Approach 3: Convolutional Neural Networks . . . 55
7.4.1 Used Convolutional Neural Network Architectures . . . 55
7.4.2 Data Models . . . 56
7.4.3 Implementation of the Convolutional Neural Networks . . . 60

8 Results 61
8.1 Difference to the Label Frame . . . 62
8.2 Frame Accuracy . . . 64
8.2.1 5-Frame Accuracy . . . 64
8.2.2 3-Frame Accuracy . . . 65
8.2.3 1-Frame Accuracy . . . 66
8.2.4 True-Frame Accuracy . . . 66
8.3 Interval and Worst Outliers . . . 68
8.3.1 90%-Interval . . . 68

8.3.2 95%-Interval 68
8.3.3 99%-Interval 70
8.3.4 Worst Outliers 70
8.4 Inference Time 71
8.5 Showcase 73
8.6 Worst Predictions for VGG16 and ResNet50 with Data Model 4_Img_1 . 76

9 Conclusion **80**

References **83**

Abstract

This thesis focuses on the automatic evaluation of puncture tests using image series obtained from a high-speed camera and employing Machine Learning techniques. The main objective is to identify the frame in the series that shows the first visible failure, referred to as the Failure Frame.

To achieve this, a two-stage system is proposed, where the first step classifies the probability of a failure at an image level. This thesis explores three classification approaches: basic image processing, Haralick features with a neural network, and twelve models that combine three Convolutional Neural Network (CNN) architectures with four input Data Models. The second stage uses segmentation via change-point detection to determine the non-failure-to-failure label change corresponding to the Failure Frame based on the classification results.

CNNs yield better results compared to the image processing and the Haralick feature approach. The best model is able to predict the Failure Frames with a deviation of plus or minus five frames at an accuracy of 90%, which corresponds to a maximum deviation of 500µs from the labeled Failure Time. The same model accurately predicts 44.1% of the Failure Frames, while another one achieves a narrower prediction range, deviating by only 24 frames when considering two outliers.

These models demonstrate the ability to identify the Failure Frame for various material behaviors. However, they encounter challenges in difficult conditions, such as insufficient lighting and reflections.

Kurzfassung

Diese Arbeit befasst sich mit der automatischen Auswertung von Durchstoßversuchen anhand von Bildserien einer Hochgeschwindigkeitskamera und unter der Verwendung von Konzepten des maschinellen Lernens. Ziel ist es, das erste Bild in der Bildserie zu identifizieren, welches Versagen der Probe zeigt, den sogenannten Versagenspunkt.

Dazu wird ein zweistufiges System vorgeschlagen, bei dem in der ersten Stufe die Versagenswahrscheinlichkeit der Probe auf Bildebene klassifiziert wird. In dieser Arbeit werden drei Klassifizierungsansätze untersucht: einfache Bildverarbeitung, Haralick-Merkmale mit einem neuronalen Netzwerk und zwölf Modelle, die drei Architekturen mit vier verschiedenen Eingabedatenmodellen kombinieren. In der zweiten Stufe wird mithilfe einer Segmentation mittels Veränderungsdetection (Changepoint-Detection) versucht, auf Basis der Klassifizierungsergebnisse der ersten Stufe den Versagenspunkt zu identifizieren.

Im Vergleich zur einfachen Bildverarbeitung und dem Haralick-Merkmal-Ansatz liefert die Verwendung von CNN-Modellen bessere Ergebnisse. Das beste Modell kann die Versagenspunkte mit einer Genauigkeit von 90% auf eine Abweichung von plus oder minus fünf Bildern vorhersagen. Dies entspricht einer maximalen Abweichung von 500µs vom vorbestimmten Versagenspunkt. Dasselbe Modell sagt den Versagenspunkt in 44,1% der Experimente ohne Abweichung genau voraus, während ein anderes Modell das Versagen in einem engeren Bereich von 24 Bildern und unter Berücksichtigung von zwei Ausreißern vorhersagt.

Die hier dargestellten Modelle zeigen, dass sie in der Lage sind, den Versagenspunkt bei unterschiedlichem Materialverhalten zu identifizieren und somit für eine Anwendung geeignet sind. Jedoch können Probleme auftreten, wenn schwierige Bedingungen wie Lichtmangel oder Reflexionen vorliegen.

Acronyms

Abbreviation	Description
AdaGrad	Adaptive Gradient
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
DS	Design Science
DSRM	Design Science Research Methodology
GLCM	Gray Level Co-occurrence Matrix
ILSB	Institute for Lightweight Design and Structural Biomechanics
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IS	Information Systems
ML	Machine Learning
MLP	Multi-Layer Perceptron
NN	Neural Network
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
TL	Transfer Learning

List of Figures

1 Design Science research framework 3
2 Design Science Research Methodology guideline 4

3 Stress-Strain curve 7
4 Stress-Strain curves for brittle and ductile materials 8
5 Impetus pendulum 10
6 Specimen clamping 10

7 Pipelines in CV 14
8 Convolution of a 2×2 kernel with an image 18
9 The GLCM from an image 19
10 The four directions for the GLCM 20
11 The perceptron concept 23
12 Feed-Forward NN 24
13 Activation functions in ML applications 25
14 Convolutional layer in a CNN 28
15 Pooling layers in a CNN 29
16 Residual and Bottleneck blocks 32
17 Dropout in a NN 38

18 The Failure Cases in the dataset 43

19 The general pipeline for the Implementation 48
20 The difference image at the Failure Frame from two consecutive images 52
21 Haralick features from cropped image 54
22 Data Model 1 Img 57
23 Data Model 3 Img 58
24 Data Model 4 Img 1 59
25 Data Model 4 Img 2 59

26 Boxplot Difference to Label Frame 62
27 Boxplot Differences with selected models 63
28 5-Frame Accuracy of all approaches 65

29 3-Frame Accuracy of the CNN models 65
30 1-Frame Accuracy of the CNN models 66
31 True-Frame Accuracy of the CNN models 67
32 90%-Interval result 69
33 95%-Interval result 69
34 99%-Interval result 70
35 Evaluation of Showcase A with VGG16 4_Img_1 74
36 Evaluation of Showcase B with VGG16 1_Img 75
37 Transparent specimen evaluation 77
38 Ice particles on specimen 77
39 Reflections and lighting issues 78
40 Two cracks at different times 79

List of Tables

1 AlexNet architecture 30
2 VGG architectures 31
3 ResNet architecture 34

4 Worst Outliers of the predictions 71
5 Mean Inference Time for the different approaches 72
6 The Worst Predictions for the best two models 76

1 Introduction

Plastics and composite materials are commonly used in many engineering applications in the automotive industry because they allow for cost-efficient and lightweight designs compared to metallic materials. Material characterization and simulation are necessary steps in the development phase to validate that occurring loads can be sustained by the component as designed. While simulation tools for metallic materials perform well, standard simulation methods for polymeric and composite materials cannot sufficiently capture the materials' behavior [9].

The cooperation partner 4a engineering GmbH (Traboch, Austria) specializes in simulating and testing various materials, including polymeric and composite materials. Material testing is essential for obtaining the material cards required for modeling simulations. The primary objective is to calibrate the parameters in the material cards for simulation to mimic the failure behavior observed during physical testing. This physical testing is carried out with the 4a impetus[1], allowing for multiple material tests, such as tensile strength and fracture tests. One specific test is the dynamic puncture test, which captures the biaxial failure behavior of materials. In the experiment, a pendulum arm with a hemispherical impactor tool is drawn to a specific angle and released onto the specimen, which fails under the load of the pendulum. The specimen is filmed from behind with a high-speed camera to determine the exact Failure Point.

With the emergence of Artificial Intelligence (AI) applications in automation processes, new solutions in the material testing field become feasible. This thesis aims to automat-

[1] https://www.4a-engineering.at/4a-impetus

ically evaluate puncture testing experiments from image data with Machine Learning (ML) methods.

1.1 Problem Statement

Evaluating a puncture test with a series of images poses noticeable challenges to a system, as different materials exhibit various characteristics when failing. Currently, the image series captured from the experiment is inspected manually, and the first image frame with a visible local failure is marked as the Failure Frame. From this point, the exact time of the failure can be calculated by dividing the number of frames by the camera's framerate and adding the time delta to the recorded time at the starting point.

Detecting the Failure Frame is relatively easy for brittle materials, as they usually fail suddenly, often between two consecutive images in the series. Ductile materials, however, show a completely different failure behavior of elongated and continuous failure within multiple image frames, complicating the detection of the actual Failure Frame (Problem 1). In addition, manual detection is tedious and time-consuming work that is susceptible to bias as individual people might perceive the state of the specimen differently, leading to inconsistent Failure Frames (Problem 2).

Therefore, the main objective is to develop an automated system that consistently detects the Failure Frame from an image series of puncture testing experiments. Because 4a engineering GmbH tests many different materials, the system has to be able to correctly identify the Failure Frame from a wide range of materials, including brittle and ductile materials. Accordingly, from the preceding problems and objectives, the research question of this thesis can be derived as follows:

How can computer vision enable systems to automatically detect the point of failure in polymers and composites using high-speed image series from puncture tests?

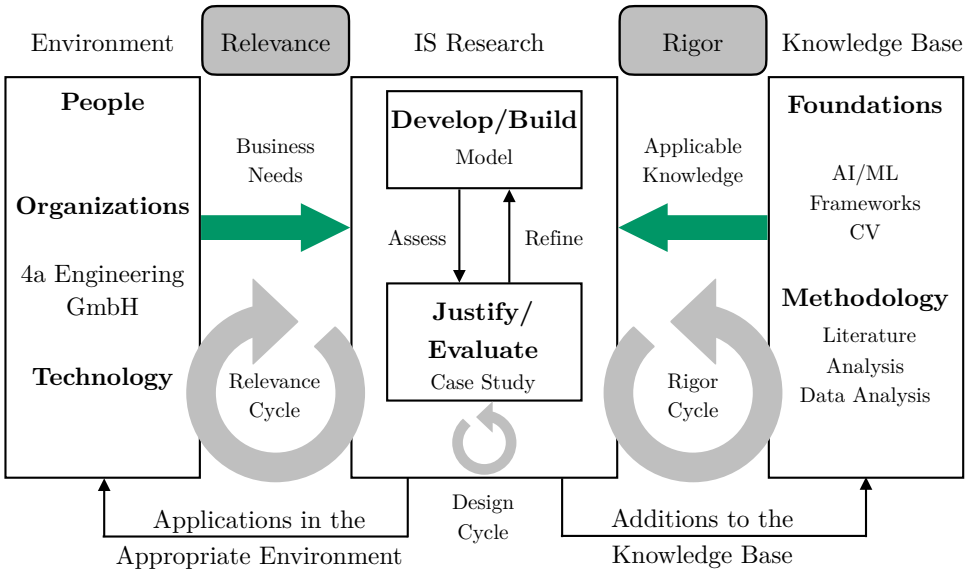


Figure 1: Design Science is a research framework enabling the structured development of artifacts based on the environment and the knowledge base in iterative cycles. (adapted from Hevner et al. [19])

1.2 Methodology and Structure

This diploma thesis aims to develop a model to detect the Failure Time of dynamic puncture tests. Therefore, a general methodology is derived to ensure the scientific approach.

Design Science (DS) is a research paradigm used in Information Systems (IS) research to develop and evaluate artifacts to solve an existing business need [19]. Artifacts are usually intangible outcomes resulting from the research process. While natural science aims to identify the truth or understand reality, DS focuses on creating artifacts serving a purpose [35]. Hevner et al. [19] have proposed a conceptual framework for research in IS, which is adapted to this research project and visualized in Figure 1. In a later publication by Hevner et al. [18], three cycles for the framework have been described to improve the understanding. These are the relevance cycle, design cycle, and rigor cycle. The relevance cycle connects the environment to the research and ensures the research project's relevance to the business [18]. To conduct a research project using DS, knowledge needs to be drawn from a knowledge base that comprises the whole research to the respective topic and consists of foundations and methods. Using this

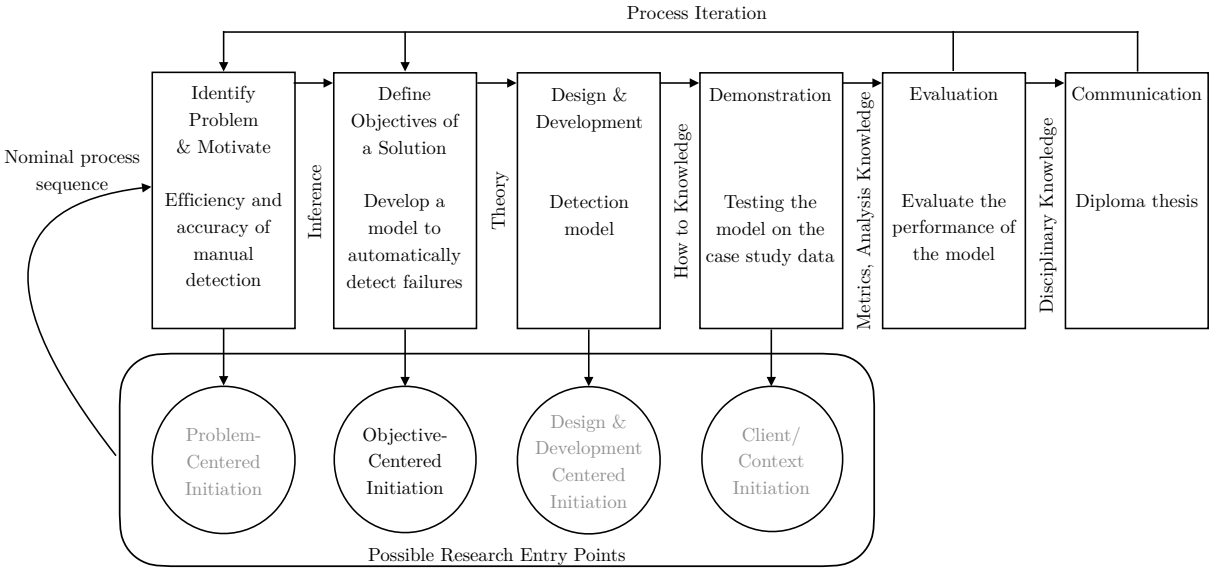


Figure 2: The Design Science Research Methodology provides a procedural guideline for conducting a research project using Design Science. (adapted from Peffers et al. [43])

knowledge describes the rigor cycle and ensures the scientific character of the research project. The design cycle of a DS project consists of two steps, the development and the evaluation of artifacts, which are carried out iterative until the designed artifact is sufficiently developed [18].

While the framework by Hevner et al. [19] provides an abstract concept, the designing of the process of a DS research project is left to the researcher. Therefore, this research project is designed according to the Design Science Research Methodology (DSRM) by Peffers et al. [43], as shown in Figure 2. The research of this diploma thesis is initiated with the objective of developing a model to predict the Failure Frame. Therefore, the objective-centered initiation is chosen from the four possible research entry points. The sequence depicted in Figure 2 starts with identifying the problems. After that, the objectives are derived, and with the application of the theory, the design and development phase is started. After obtaining a model, the capabilities are demonstrated and then evaluated. This diploma thesis informs about the findings and outcome of the research project.

This thesis is structured as follows. In Chapter 2, a brief introduction to material testing is given. Testing procedures are detailed, with a particular focus on the puncture test. Furthermore, the evaluation of material tests is also explained to enhance the understanding of the overall objective of this thesis. Chapter 3 provides a state-of-the-art overview of ML and Computer Vision (CV), explaining general concepts relevant for this thesis. This includes basic information on image processing, the extraction of features, Neural Networks (NNs) and Deep Learning (DL) methods, and training of ML models. These two chapters build the foundations of the knowledge base.

Then, Chapter 4 focuses on literature in the context of failure classification and crack detection, drawing from a previously conducted literature review that has been carried out in the course of a project work [44]. Therefore, only the most important results from the literature are summarized and stated. In the same project work, the dataset provided by 4a engineering GmbH has been analyzed using an exploratory data analysis. These findings are mentioned in Chapter 5 and build the context through the knowledge base for the practical use case.

Chapter 6 connects the objective of the thesis to the findings from the literature and the data analysis and, combined with the information of the previous chapters, derives three approaches for the implementation in the use case. The following Chapter 7 details the implementation of these approaches before the results are presented in Chapter 8.

This thesis is concluded in Chapter 9, where all essential findings are summarized, the scope is reevaluated, and an outlook and promising entry points for further research are mentioned.

2 Introduction to Material Testing

This chapter briefly introduces material testing and stress-strain curves before the specific puncture test by 4a engineering GmbH is detailed. The information obtained within this chapter enhances the knowledge base and promotes an improved understanding of the advantageous application of an automated evaluation system.

Materials in most engineering applications are exposed and have to sustain external loads without failing. Therefore, material testing characterizes material properties, such as the ultimate tensile strength and Young's modulus, which allows for the efficient designing of parts. Stress-strain curves are commonly used to calculate these material parameters and are explained in the following.

2.1 Force-Displacement and Stress-Strain Curve

In a tensile test, a material specimen is subjected to an axial force that changes the length of the specimen [5]. The force and displacement are measured until the specimen fails and can then be arranged to create the force-displacement curve [13]. However, this curve is dependent on the geometric properties of the test specimen, which is unwanted as independent material characterizations are required for designing parts. There, the maximum stress that a specific area can tolerate is relevant. Therefore, stress and strain are used to obtain the parameters of the material [13]. The engineering stress

$$\sigma = \frac{F}{A_0} \tag{1}$$

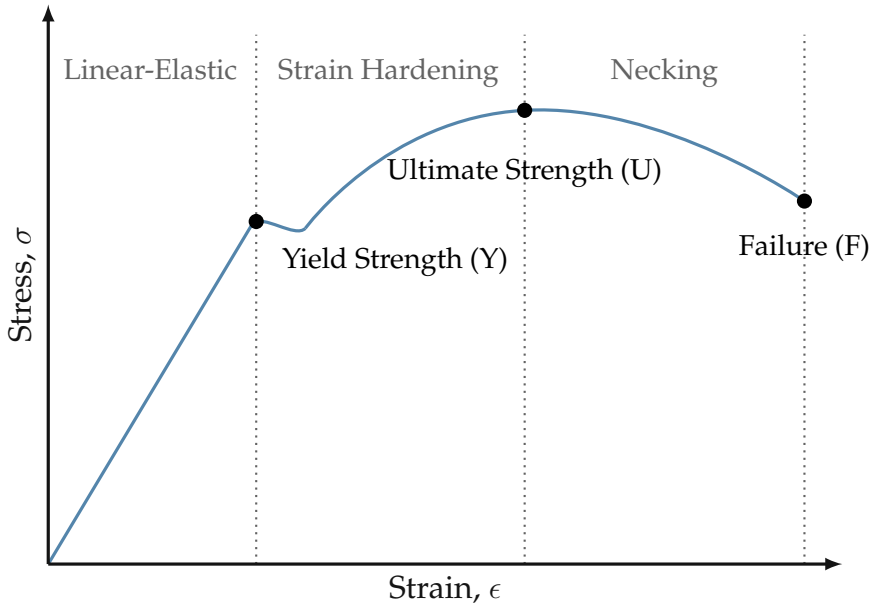


Figure 3: The stress-strain curve results from a tensile test, where important material parameters, such as Young's modulus, yield strength, and ultimate strength, can be extracted. (adapted from [40])

is defined as the Force F applied by the machine divided by the original area of the cross-section A_0 and usually measured in N/mm^2 [13]. The engineering strain

$$\epsilon = \frac{\Delta L}{L_0} \tag{2}$$

is the elongation of the specimen ΔL divided by the original length L_0, which results in a dimensionless value [13]. Applying this to the force-displacement signal, the engineering stress-strain curve of a material can be derived, as shown in Figure 3. Ductile and brittle characteristics of materials can be distinguished by the stress-strain curve.

Ductile materials, such as steel and many alloys, have three different regions that can be separated [5]. The first region is linear-elastic, where the stress and strain are proportional [4, 5]. In this region, Hooke's law is valid, and the Young's modulus [4]

$$E = \frac{\sigma}{\epsilon} \tag{3}$$

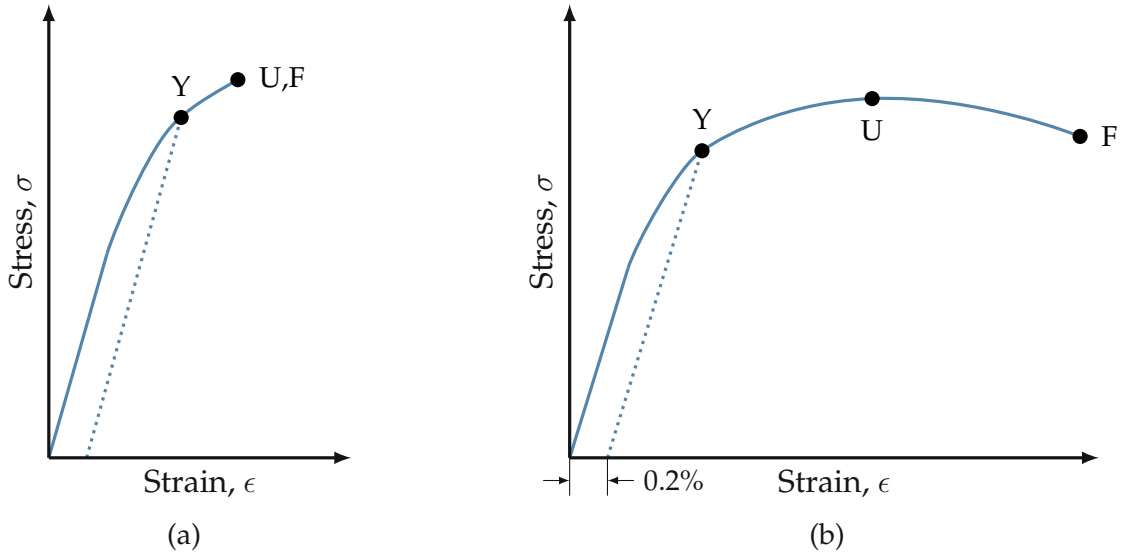


Figure 4: The stress-strain curves for brittle (a) and ductile (b) materials differ significantly, as brittle materials lack the necking phase. The dotted lines are used to obtain the yield strength using the 0.2% method, which determines the yield strength at the intersection of the dotted line and the stress-strain curve. (adapted from mechanicalc.com)

can be derived from the stress σ and strain ϵ, equating to the slope of the curve [4]. The yield strength limits the linear-elastic region and marks the maximum stress that can be applied to the material without permanent deformations [5, 13]. The yield strength is the maximum tolerable stress in the designing phase for many engineering applications. The strain-hardening phase marks the second region in the stress-strain curve, where the maximum stress reaches the ultimate strength point [5]. After that, the diameter of the specimen decreases locally, and less force is required to strain the specimen until the fracture occurs [13]. The stress at the rupture is referred to as the breaking strength [5]. Figure 4 shows two characteristic stress-strain curves for the different behavior of ductile and brittle materials. The yield strength is designated by the letter "Y," while "U" indicates the ultimate strength, and "F" is used to denote the failure of the sample.

Brittle materials, as illustrated in Figure 4a, show no to minimal changes in the slope of the curve and sudden rupture [5]. The rupture occurs at lower strains compared

to ruptures of ductile materials, and without the different regions, the failure strength equals the ultimate strength [5].

Like many polymers, very ductile materials have no distinct yield point but a progressive strain with a non-linear stress dependency [5]. This behavior is shown in Figure 4b. To obtain the yield strength for these materials, the 0.2% offset method can be used [5, 4, 13]. There, the elongation of 0.2% is fixed on the strain axis, and a line parallel to the initial stress-strain curve is added [13]. At the intersection of the parallel line and the stress-strain curve, the 0.2% yield strength is obtained, as shown in Figure 4b [4].

The obtained parameters are vital for engineering purposes, and many other material tests have been developed.

As this thesis focus is not on material testing, this chapter only provides a brief introduction to the topic to enhance the context. Only the puncture test of 4a engineering GmbH, which has been used to obtain the dataset for this thesis, is detailed in the following.

2.2 Experimental Setup and Testing Procedure

The dynamic puncture test by 4a engineering GmbH is carried out on material samples using their impetus pendulum testing machine [1]. It enables five different tests with different setups and the machine is shown in Figure 5 [1]. The dynamic puncture test is specified in DIN EN ISO 6603-2 [28] and describes a testing procedure where an impactor is released onto a specimen vertically. The impetus has been developed around this standard but does not strictly follow it, as the pendulum arm impacts the specimen at the resting position in the horizontal direction [1].

The main focus of the puncture test is to determine the impact behavior of the specimen [27]. Therefore, a specimen is clamped within two supporting plates (specimen clamping) before being inserted into the counter bearing, as shown in Figure 6 [1]. The

Figure 5: The impetus pendulum by 4a engineering GmbH is used for multiple material tests, which form the basis for the material card development. (retrieved from https://www.4a-engineering.at/4a-impetus)

pendulum arm is then deflected to a specific angle to control impact speed and energy before being released onto the specimen. The applied impactor with a hemispherical tip punctures the specimen, which fails under the load. An external highspeed camera is placed behind the specimen, as shown in Figure 5, and used to capture image series of the puncture test [1]. These image series from multiple experiments build the dataset for this thesis.

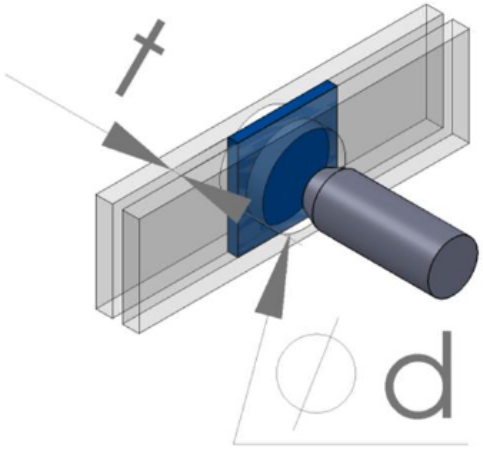


Figure 6: The specimen is clamped between two metal plates that support the specimen during the puncture test. The impactor hits the specimen in the center and perpenticular. (retrieved from [1])

Several parameters are sensed during the puncture test. The head of the pendulum arm features an acceleration sensor capable of detecting up to 400g (gravitational constant), and the counter bearing that holds the specimen contains an acceleration sensor with a maximum capability of 25g. [1]. The arm of the pendulum can be fixed and released using an electromagnetic disc brake, and together with an angle sensor which is used to track the pendulum, can control the impact velocity, which is $3m/s$ as standard [1]. Other sensors, such as humidity and temperature sensors, allow for the detection of changing climate conditions [1].

With these data obtained, the signals from the angle sensor and the two acceleration sensors are inspected and build the basis for further analysis. For the force-displacement curve, the force

$$F = m_{\text{pend}} a_{\text{pend}} \tag{4}$$

is calculated using the mass m_{pend} and acceleration a_{pend} of the pendulum [1]. The distance

$$s = s_0 + \frac{(\alpha_0 - \alpha_1)\pi}{180} L_{\text{pend}} \tag{5}$$

can be derived from the differences in the angle sensor α_0 and α_1 multiplied with the length of the pendulum L_{pend}, from a starting point s_0 [1]. The displacement is the distance starting from the contact point of the impactor and the specimen. With the displacement obtained, the force-displacement curve can be plotted and analyzed. Stress-strain curve can be obtained since the force-displacement curve is directly correlated. The Failure Point of a material is given by the maximum stress that can be sustained without local necking, referred to as ultimate strength in Figure 3.

To determine the exact point of failure, the image series is used. In a manual process, the images are inspected, and the image containing the first visual local failure is selected as the Failure Frame [1]. From the frame number of the Failure Frame, the exact timestamp is determined and used as the breaking point [1]. The objective of the thesis is the automated detection of this Failure Frame from the image series. The methods

used to achieve this heavily rely on state-of-the-art CV and ML concepts, which are described next.

3 State-of-the-Art: Computer Vision and Machine Learning

In this chapter, the theoretical basis for this thesis is described by introducing concepts from image processing, AI, ML, and CV. CV is a research area that uses AI and ML concepts to recover details from image data [54]. Shapiro and Stockman [50] have defined a general goal for CV:

> "The goal of computer vision is to make useful decisions about real physical objects and scenes based on sensed images."

This statement highlights the highly complex nature of CV as any interpretation and decision-making requires extensive information about the situation. Humans can obtain information by sensing the surroundings visually [50]. The combined ability to connect this information to abstract contexts and scenes is straightforward for humans. However, computers require a representation of the object to process the information [12]. The representation of an object is built up of features, which form its knowledge body [12]. Specific features could be height, age, and eye color to represent a human. With only these traits captured, the machine can only process that information, as the sum of all features limits the representation of the object. Therefore, selecting the right features is vital in any ML and CV approach.

To enhance the understanding of state-of-the-art CV methods, it is important to review the beginning of CV as a research field. The simplest form on CV has been developed in the 1970s when digital image processing emerged [54]. Digital image processing

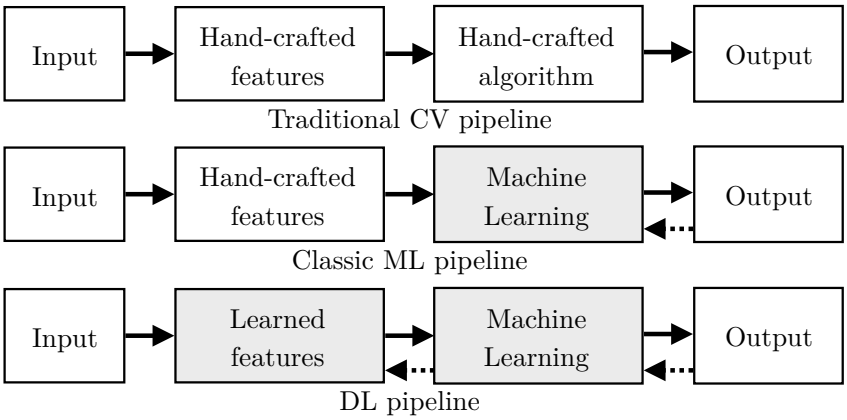


Figure 7: The three pipelines show different levels of technology in CV approaches. (adapted from [54, 12])

uses simple concepts such as image transformation, masking, and filtering [54, 50]. One particular approach of digital image processing for detecting motion in image series is described in more detail in Section 3.1. As shown in Figure 7, the traditional CV pipeline translates an input into designed features before a carefully designed algorithm transforms them into an output. All steps are hand-crafted, but the calculation is done by the computer.

With the advances in computer technology and AI, the focus has shifted towards ML, which allows for pattern extraction and information generation, therefore enabling simple learning algorithms [12, 54]. The representation of the image data is still engineered by hand, but the ML model directly maps the features to the output, learning only the mapping itself. This is shown in the second level in Figure 7.

The third level in Figure 7 visualizes the DL pipeline, where the model learns the representation and mapping, providing an end-to-end approach [12]. The features learned in DL can also contain abstract compound features that are not intuitive for humans.

The three classification approaches presented in this thesis each correspond to a different CV pipeline, as mentioned above. Therefore, in the following, features and the extraction of features from images are described before relevant ML and DL concepts are presented.

3.1 Image Processing

To represent an image in a computer system, the pixel values are typically composed in a two-dimensional matrix of $width \times height$ pixels containing the pixel intensities. Typically, when color information is stored, each color channel is represented by one matrix, resulting in a three-dimensional representation of $width \times height \times channel$. Digital processing of images uses this representation as a basis to which manipulations can be applied.

Image processing is almost always used in CV pipelines, as representations of images have to be converted to allow a suitable application [54]. Simple forms of this are the correction of color, exposure, and noise reduction, leading to an improved representation of the captured object in the image [54]. The most basic processing transformations are referred to as point operators and the simplest ones are applied on the pixel level, meaning that the pixel is altered without considering other pixels [54]. Point operators are briefly described because the first approach in this thesis is based on that concept.

3.1.1 Point Operations

Generally, point operations calculate an output by applying a function to an input image. Commonly, to obtain the output

$$g(\boldsymbol{x}) = a(\boldsymbol{x})f(\boldsymbol{x}) + b(\boldsymbol{x}), \tag{6}$$

the input image $f(\boldsymbol{x})$, where $\boldsymbol{x}$ refers to the pixel locations, is multiplied by a parameter $a(\boldsymbol{x})$, and a parameter $b(\boldsymbol{x})$ is added [54]. The parameters in Equation (6) are variable on the spatial dimension, which means that values from other pixels can affect the output of a pixel [54]. Advanced applications can utilize the spatial dependence of the parameters to modify the input image selectively [54]. However, in their simplest

form, these parameters are constants and, therefore, independent of the input image. This can be used for contrast and brightness adjustments of images [54].

Some problems in image processing translate to detecting motion from an image sequence [50, 29]. There are different types of settings when characterizing motion in a series of images, as the motion can stem from the movement of the camera, the captured object, or a combination of both [50]. In the case of a puncture test, the camera is fixed, and only the specimen, as the captured object, is subjected to motion. When observing the gathered images as they progress over time, the object moves by some pixels, resulting in the information being shifted in the representation. The critical information that needs to be extracted is the set of pixels that show significantly different values compared to the image before [45]. Image subtraction can reveal the relative motion between two images by calculating the pixel difference [29]. The difference image

$$\mathbf{D} = \mathbf{I}_i - \mathbf{I}_{i-1} \tag{7}$$

compares the pixel values of the current image $\mathbf{I}_i$ and the previous image $\mathbf{I}_{i-1}$ and calculates the difference [45, 29, 50]. As a result, drastic changes in the pixel values are detected, while areas with few changes contain low pixel values and are noisy. Image subtraction is typically performed on images that share significant parts of the image [42]. A technique to enhance differences is the thresholding technique, where a specific threshold value is defined, and all pixels containing values smaller than the threshold are set to zero [54].

3.1.2 Digital Image Filtering - Convolution

Another set of operations in image processing uses more than one pixel to calculate the output cell. Neighboring cells are considered to filter images for sharpening, blur, or edge detection purposes [54, 12]. To describe the filtering process, the convolution has to be introduced.

The convolution in image processing uses an input image and a kernel to generate a feature map as the output [12]. An intuitive approach to the convolution is that the kernel selects and transforms the neighboring cells to determine the output. The kernel is a two-dimensional matrix that is convolved with the original image. In each step, the overlapping pixels are multiplied with the corresponding values in the kernel, which are then summed to determine the pixel's value in the output image, the feature map [12, 54]. Then, the kernel is moved to the next position, and the process is repeated. The stride parameter determines the number of pixels the kernel is moved between each step [54].

In the mathematical definition of the discrete two-dimensional convolution, the output [12]

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{8}$$

is calculated by convolving the input image I with the kernel K, as denoted with the $*$ operator, which corresponds to the summation of the products of I and K in every pixel [12]. The negative sign in the formula stems from the definition of the convolution in one dimension and is also required in two dimensions to preserve the commutative property [12]. This negative sign can be interpreted as flipping the kernel along both axes around 180°. The operation without flipping the kernel is called correlation. However, as the filter is often symmetric or specifically engineered to detect edges, the flipping of the kernel is sometimes not done [12]. Therefore, in the following, the kernel flipping is assumed but not actually part of the description.

The convolution of a 2×2 kernel on an input image with stride one is shown in Figure 8. Different kernels achieve different effects on the input image. A kernel consisting of zeros with a one at the center will return the input image, as only the center pixel is weighted. Other typical kernels, such as the Gaussian kernel, are frequently used to smooth images [54].

During convolution, the kernel moves across the input image, calculating the output. However, due to the limitations of the kernel's movement, the edges of the image

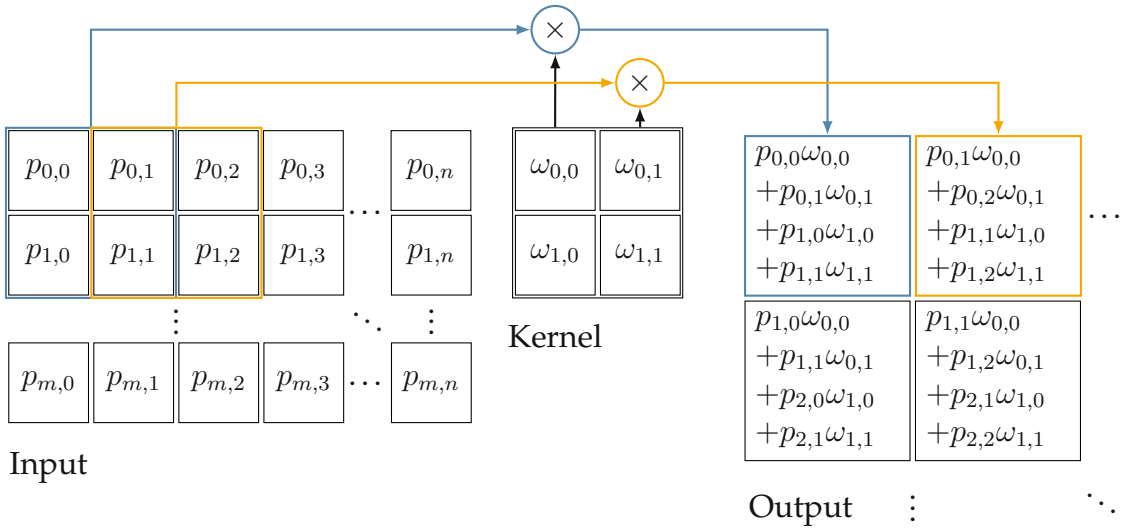


Figure 8: The convolution of a 2×2 kernel with an input image produces an output image that is influenced by the values of the kernel and the close proximity values in the input image. (adapted from [12])

remain unevaluated, shrinking the dimensions of the output. This may be undesirable as image size can be important. To resolve this problem, padding techniques have been devised by developers to increase the input image with an additional pixel boundary. This results in a greater area for kernel shifting [54]. These padding methods include zero padding, which sets all pixels outside the input image to a value of zero, and constant padding, which assigns all pixels to a specific value [54].

The convolution is, therefore, an important tool to generate feature maps, which can be processed further in a system by either a hand-crafted algorithm or a ML algorithm. In the following, the extraction of texture features from images is described, as they can also be used to represent an image.

3.2 Texture Features - Haralick Features

Texture, as defined by Haralick et al. [15], corresponds to a property of surfaces, capturing the structure and spatial relationships. Intuitive texture forms are, for example, soft and rough to describe surface characteristics [15]. Whereas humans are trained to identify texture quickly, computers have difficulties identifying this information. To

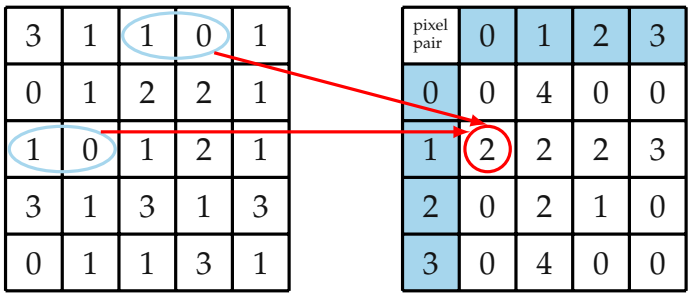


Figure 9: The GLCM is calculated by identifying the number of occurrences for each pair of pixel values in the input image. This image consists of only four different pixel values, and the direction in this example is 0°. The pixel pair (1,0) occurs two times in the image, which is marked in the highlighted cell in the GLCM. (adapted from [59])

allow computers to access information on texture from images, Haralick et al. [15] developed texture features, which can be used for object identification and classification [15, 36]. They propose 14 features computed from a gray-tone spatial-dependence probability-distribution matrix, commonly known as Gray Level Co-occurrence Matrix (GLCM) [15]. To obtain the Haralick features for an image, the GLCM needs to be obtained first, which is explained below.

As a texture analysis approach, the GLCM is a method to capture the relation of adjacent pixels in a spatial dimension and is widely used for many purposes [38, 33]. The matrix width and height correspond to the range of the pixel values in the image, usually with 256 different values a pixel can contain, the GLCM consists of 256 rows and columns [38]. Each cell in the GLCM defines a specific pair of pixel values (row and column index) [15, 33, 38]. When calculating the value of the cell in a GLCM, the original image is searched for occurrences of the pair of pixel values being adjacent to one another. This calculation depends on the direction, and the order of the value-pairs is also important, resulting in the GLCM being non-symmetric. This calculation of the GLCM is shown in Figure 9. That means that the cell in the second row and the third column contains the total number of occurrences where gray levels two and three are in adjacent pixels.

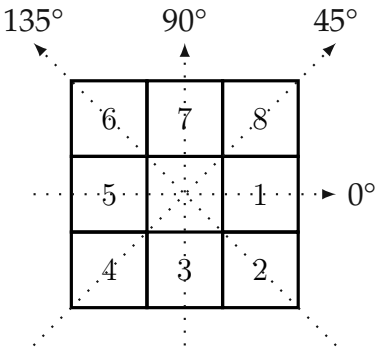


Figure 10: The four directions that connect adjacent pixels can be used to calculate the GLCM. (adapted from [15])

The GLCM can be calculated in four directions, which are horizontal, vertical, and diagonally at 45° and 135° [15]. These directions connect all eight surrounding pixels to a center pixel, as visualized in Figure 10. Four GLCMs are calculated separately, with one for each of the four directions. The Haralick features are then calculated from these GLCMs. 14 Haralick features can be extracted from each GLCM, according to the calculations of Haralick et al. [15]. When considering all four GLCMs, a total of 56 features can be obtained. Whereas the 14 individual Haralick features capture an image's texture, they admit that it is tough to understand which characteristic is represented by the individual features. Therefore, the Haralick features are stated below, but without further detailing and calculation formulas, as they can be found in the original publication [15].

The 14 Haralick features are [15]:

1. Angluar Second Moment
2. Contrast
3. Correlation
4. Sum of Squares: Variance
5. Inverse Difference Moment
6. Sum Average
7. Sum Variance
8. Sum Entropy
9. Entropy
10. Difference Variance
11. Difference Entropy
12. Information Measure of Correlation 1
13. Information Measure of Correlation 2
14. Maximal Correlation Coefficient

3.3 Segmentation vs. Classification

Many different tasks can be solved using ML algorithms. In this thesis, the concepts of classification and segmentation are important, as both methods are used in the use case. For classification tasks, the model generates a prediction regarding the category of an input. For example, in image classification, the model is exposed to an image containing an object, and the algorithm returns a prediction for the category of this object. This can be a discrete value corresponding to a predefined class or a probability distribution for multiple classes. [12]

Segmentation can be used in multiple contexts, such as image segmentation, where objects are separated on a pixel-level basis. Segmentation is also used in signal processing, as signals often show different sections with specific characteristics. Determining the different signal regimes is essential in applications such as weather forecasting or machine signal monitoring for quality control [20]. A change in the vibration signal can indicate machine failures or quality issues. The concept of detecting changes in a signal translates to the changepoint detection problem described in the following.

3.4 Changepoint Detection

Changepoint detection aims to identify the location where changes of specific parameters in a signal occur [57]. This task can be divided into online and offline methods, with online referring to real-time changepoint detection and offline detecting changepoints after the whole signal has been retrieved. This is also often described as signal segmentation, with the task of selecting a model to fit the data. The three core elements in the changepoint detection system are the cost function, the search method, and the constraint [57].

The constraint can either be a known number of changepoints or the segmentation without previous knowledge of the number of changes. The cost function is used as

a criterion that measures the model's fit and returns a cost for samples with a suboptimal fit. Cost functions can be used on parametric data containing a finite number of parameters or non-parametric data. This thesis focuses on parametric data; therefore, only these are detailed in the context of changepoint detection [57]. Also, the main goal is to detect shifts in the mean of a signal, which leads to a maximum likelihood estimation with a cost function that calculates the quadratic error loss [57]

$$c(y_I) = \sum_{t \in I} \|y_t - \bar{y}\|_2^2 \tag{9}$$

of the signal y with the mean $\bar{y}$ on the interval I [57]. With the loss function specified, the search method is determined in the next step. In changepoint detection, an optimal or approximate search can be conducted. The optimal search returns the exact solution of the segmentation that achieves minimal cost by calculating the loss for every possible segmentation [57]. This results in high computational requirements, which can be limited by applying approximate methods. Important approximate segmentation methods are binary segmentation, bottom-up segmentation, and window-sliding segmentation [57].

The changepoint detection in this thesis can be achieved with the optimal search method, as the sequence is short enough, and the number of changes is fixed to one. Therefore, only this method is considered in the following.

3.5 Deep Learning

DL is a ML concept that leverages learning of representations in different stages, as complex representations can be learned from simpler ones [12, 54]. A car in an image can contain simple representations such as edges and curves, which combined form more sophisticated concepts such as wheels, windows, and mirrors. To achieve this, the basic idea of the neuron and NNs is introduced in the following.

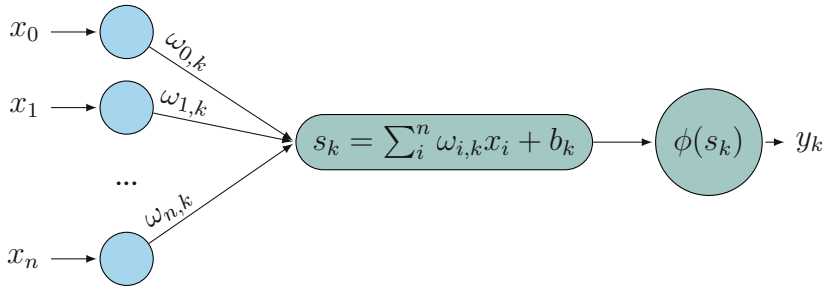


Figure 11: The perceptron transforms inputs by applying weights and bias before a selected activation function returns the output [54, 48].

3.5.1 Neuron and Fully Connected Neural Network

The perceptron, also referred to as neuron, has been described by Rosenblatt et al. [48] and marks the smallest computational unit in a NN. The development of the neuron has been inspired by the biological concept of neurons in the brain. The perceptron k, as initially intended, receives the inputs x_i and multiplies them with weights $\omega_{i,k}$ before a bias b_k is added to obtain the weighted sum [54]

$$s_k = \sum_i^n \omega_{i,k} x_i + b_k. \tag{10}$$

To enable the perceptron to learn non-linear functions, a non-linear activation function ϕ is applied to the weighted sum s_k to form the output y_k as shown in Figure 11. The activation functions are essential to the performance of NN and are described in Section 3.5.2. The neuron can be trained to return the desired output, where it adapts its weights and bias to fit the data from the training [12].

Multiple perceptrons can be organized into layers and interconnected to form a feed-forward NN. Each perceptron in a layer receives the output of any neuron of the previous layer and returns an output, which is then processed by neurons of the next layer. The number of layers in a NN characterizes the depth of the network, and the number of neurons within a layer corresponds to the width [12]. This results in a NN, which is often referred to as Multi-Layer Perceptron (MLP) and shown in Figure 12.

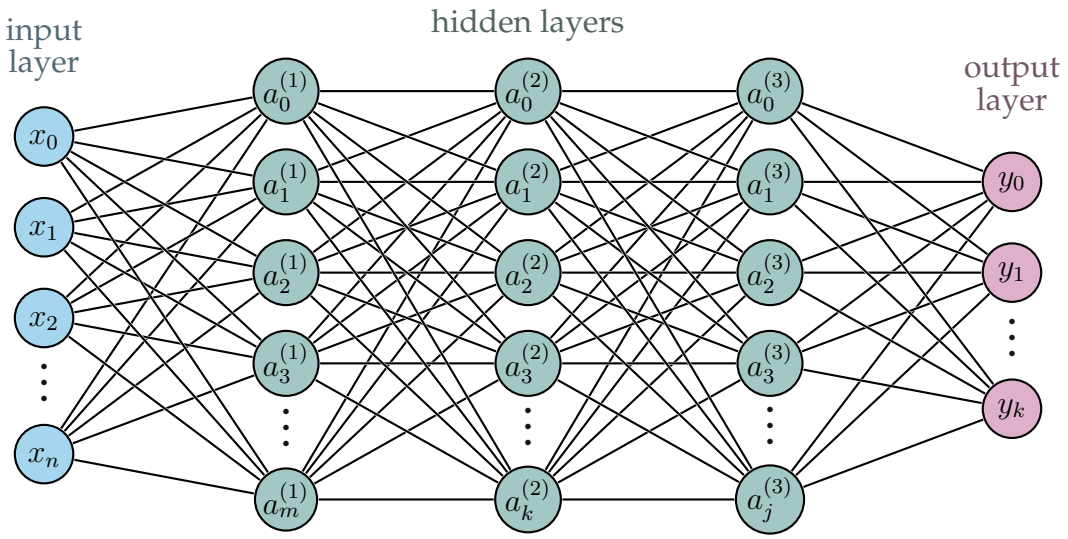


Figure 12: The feed-forward NN is a stack of layers consisting of multiple neurons. These networks typically consist of an input, an output, and one or many hidden layers. [48, 12] (adapted from [39])

Feed-forward NNs are universal approximators, meaning that a network with at least one hidden layer and sufficiently enough neurons can represent any function to any desirable exactness [22, 12]. This universal approximation theorem, however, does not describe how large the network needs to be [12]. Also, different training procedures can be used, so the optimization algorithm can lead to non-optimal learning results. Usually, more layers of hidden neurons are used to form deeper networks, as they require fewer neurons to approximate the function [12].

3.5.2 Activation Functions

This section covers activation functions in more detail. As mentioned above, the activation function transforms the weighted sum of a perceptron to form the output. The specific function that is applied to this sum allows a network to learn different behaviors. As categorization, either linear or non-linear activation functions are considered. However, linear activation functions are not able to represent non-linear behavior, and therefore, the MLP cannot learn these functions [12]. Non-linear activation functions provide this characteristic, enabling the network's universal approximator property

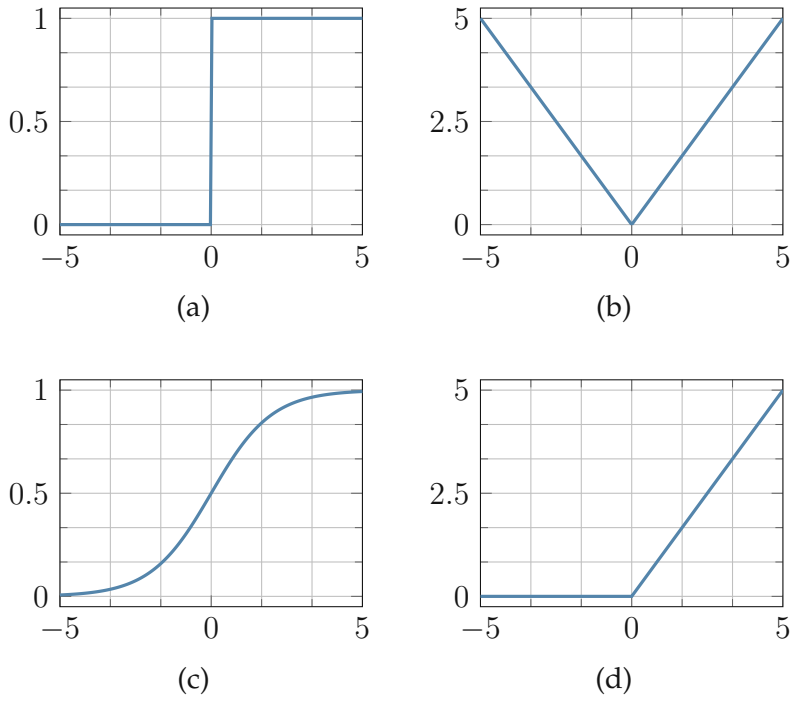


Figure 13: The Heaviside (a), absolute (b), sigmoid (c), and ReLU (d) activation functions are commonly used in ML and introduce non-linearity into the system. [3]

[3]. They transform the output of a neuron and feed the non-linear output to the neurons in the next layer or the output itself when considering the activation in the last layer of a network.

Common activation functions are shown in Figure 13. The Heaviside step function (Figure 13a) returns one if the input value is positive and zero otherwise and is only used in the last layer of a network [3]. The absolute function in Figure 13b, however, retains all positive values, whereas negative values are transformed into the absolute value [3]. The sigmoid function shown in Figure 13c is especially useful when the output should be within the range of zero and one with a continuous transition. This activation function is, therefore, often applied in binary classification tasks, as the output value can be interpreted as a probability for a class. The softmax activation function, which is similar to the sigmoid activation function, is often used in the last layer of a classification model with multiple classes, as it normalizes the probability and returns a probability for each class [12].

In DL, the Rectified Linear Unit (ReLU) activation function, as shown in Figure 13d, is most commonly used due to its simplicity and because it is less prone to the vanishing gradients problem [11, 3]. The

$$ReLU(x) = \max(0, x) \tag{11}$$

function returns the exact value for input with positive values and zero for negative inputs. Opposed to the sigmoid function, this results in a large proportion of the neurons returning a zero value, which enhances sparsity in the network and allows for faster and simpler computation [11].

3.5.3 Loss Functions

Loss functions are used to determine the deviation of the predictions to the labels [54]. When training a NN, the objective is to minimize a predefined loss function by adjusting weights and biases, which results in the model fitting to the data [54]. As the tasks of NN vary, the loss functions can be divided into categories corresponding to regression and classification. Whereas regression tasks require loss functions, such as the mean squared error, for classification, loss functions, such as binary cross-entropy and categorical cross-entropy, are used [58]. As the thesis covers a classification task, only this category of loss functions is detailed.

The binary cross-entropy or log loss is widely used for classification tasks with two output states [25, 58]. It calculates the loss [21]

$$J_{\text{bce}} = -\frac{1}{M}\sum_{m=1}^{M}[l_m \log(h_\theta(x_m)) + (1 - l_m)\log(1 - h_\theta(x_m))] \tag{12}$$

where M is the number of training samples, l_m specifies the actual label for the example m, x_m the corresponding input, and h_θ the model with weights θ of the network [21]. When the target value of the sample is one, and the model predicts 0.8, the pre-

diction is 20% wrong and triggers a penalty, which is used for the next training step [21]. The penalty value then equates to $-\log(0.8) = 0.097$ for this example, and in the case of a perfect prediction on one, the penalty results in $-\log(1) = 0$. Therefore, the binary cross-entropy loss function heavily penalizes predictions that deviate substantially from the true label, which allows for good training performance in binary classification tasks. Categorical cross-entropy is similar to binary cross-entropy but used for classification tasks with more than two categories.

3.6 Convolutional Neural Networks

In CV, NNs typically receive image data in a grid structure as input. These grids contain the values of each pixel in the image, which results in an extremely large number of individual inputs to a network. A fully connected MLP would require layers with enormous numbers of neurons, which is computationally very expensive. Convolutional Neural Networks (CNNs) are building up from the concept of convolution, where not the individual pixel values are connected to the neurons, but instead, features are extracted from the image by convolutions and learned by the model [12]. Usually, additionally to convolutional layers, CNNs also contain pooling layers that reduce the size of the output [14, 12].

3.6.1 Convolution and Pooling Layers

Convolution layers consist of a set of kernels that are convolved with every area of the input according to the stride specified [54, 12]. The used kernels can vary in size, and the main concept of the CNN is that the kernels are learned during training. This allows the model to extract useful features, even though they might not appear relevant when inspected by a human. CNNs can contain multiple convolutional layers, where the feature maps from one layer are used as input for the next convolutional layer [54], as shown in Figure 14. One important advantage of CNNs is that the weights of the

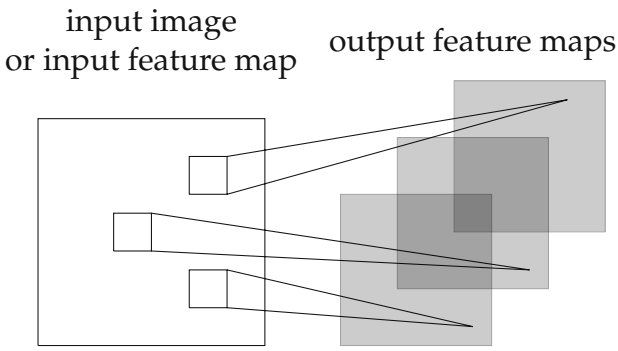


Figure 14: The convolutional layers in a CNN extract feature maps from an input image or a previous feature map. Each kernel is convolved with the input and extracts a feature map with the same parameters in the kernel. The number of kernels in a convolutional layer directly determines the number of obtained feature maps, which translates to the dimensionality of the output. (adapted from [53])

convolutional kernels remain the same even when the kernel is applied to different regions of the input [54]. This is referred to as weight or parameter sharing and reduces the number of parameters in the system [54, 12]. Another property is sparse interactions, meaning that in a convolutional layer, only a specific area covered by the kernel contributes to the output of a unit [12].

Pooling layers modify the output to reduce the size and account for shifts in the feature maps [14]. There are many pooling methods, but the most commonly used are max pooling and average pooling [12, 54]. When applying max pooling, a window of a specified size is shifted over the input and extracts the maximum value within the window at each area [12, 54]. Average pooling uses the same process, but instead of the maximum value, an average of all values is calculated [14]. These values are then building the output, which is reduced in size [14]. One main intention is that not the location of the features in the feature maps is important, but the presence [14]. Figure 15 shows the general approach of pooling layers, where an area within an input is condensed to a single value in the output.

In the following, the convolutional layer is used for the convolution and the detection via a non-linear activation function [12]. CNNs can contain many convolutional layers, which allows for more complex features to be detected, and also, with every layer, the

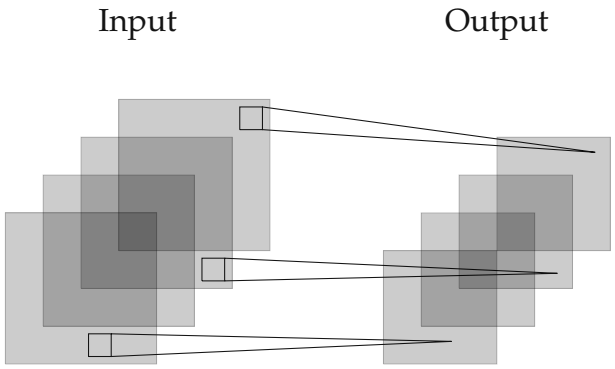


Figure 15: The pooling layers in a CNN are used to condense information of feature maps. Similar to a convolution, the pooling operation slides a window over an input and determines a single value for a given step and area of the window on the input. The method, such as maximum or average, can be designed individually for different layers to determine the output value from the input values. (retrieved from [53])

observed area of the original input enlarges [12]. After the convolutional layers, the output can be fed into fully connected layers, which are then used to perform the DL task [14].

3.6.2 Network Architectures

As there are many possible combinations of the different layers and parameters, various architectures for CNNs have been developed. Many of them have emerged because of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [49], which led to rapid advances in the field of CV and especially object detection. Three architectures AlexNet [32], VGG [51], and ResNet [17] demonstrate different approaches and are described below.

AlexNet

Krizhevsky et al. [32] have proposed the AlexNet architecture in 2012 to classify images in the ILSVRC-2010 contest, where it achieved higher performance than other architectures. This network architecture consists of eight layers, five convolutional and three fully connected layers at the end of the network [32]. Table 1 shows the architec-

Table 1: The AlexNet architecture contains five convolutional layers, three dense layers and classifies images of 1000 classes. (adapted from [32])

AlexNet
Input (224×224×3)
11×11 conv-96
max pool
5×5 conv-256
max pool
3×3 conv-284
3×3 conv-284
3×3 conv-284
max pool
FC-4096
Dropout 0.5
FC-4096
Dropout 0.5
FC-1000 (softmax)

ture in a visual representation. The network requires the input to be an RGB image of size $224 \times 224 \times 3$ before the first convolutional layer is applied. The first convolution layer uses 96 kernels with a size of 11×11 pixels, which are convolved with a stride parameter of four and a padding that adds zero values at the edges [32]. The second convolutional layer consists of 256 kernels, which are smaller with a size of 5×5 [32]. The third and fourth convolution layers each have 384 kernels with a size of 3×3, and the fifth convolutional layer uses the same kernel size but consists of 256 kernels [32]. The ReLU activation function is applied after each convolutional layer. Max pooling is applied to the first, second, and fifth convolution layers [32]. Then the output is fed into three fully connected layers with 4096 neurons in the first two layers and 1000 neurons in the last layer, such that every output neuron corresponds to one of the 1000 different classes [32]. Dropout layers, which will be explained in Section 3.7, reduce overfitting and are inserted after the first and second fully connected layers [32]. The softmax activation function of the last and, therefore, the output layer returns a probability distribution for the 1000 classes [32].

Table 2: The different VGG architectures consist of the same layers, and only the number of the convolutional layers increases for deeper VGG networks. (adapted from [51])

VGG11 11 weight layers	VGG16 16 weight layers	VGG19 19 weight layers
Input (224×224×3)		
3×3 conv-64	[3×3 conv-64] ×2	[3×3 conv-64] ×2
maxpool		
3×3 conv-128	[3×3 conv-128] ×2	[3×3 conv-128] ×2
maxpool		
[3×3 conv-256] ×2	[3×3 conv-256] ×3	[3×3 conv-256] ×4
maxpool		
[3×3 conv-512] ×2	[3×3 conv-512] ×3	[3×3 conv-512] ×4
maxpool		
[3×3 conv-512] ×2	[3×3 conv-512] ×3	[3×3 conv-512] ×4
maxpool		
FC-4096		
FC-4096		
FC-1000 (softmax)		

VGG

The VGG-Net originates from the Oxford Visual Geometry Group (VGG) [54], uses also an RGB image of size $224 \times 224 \times 3$ and has been proposed by Simonyan and Zisserman [51] as an entry in the ILSVRC 2014. Opposed to the AlexNet, the VGG-Net only uses small 3×3 filters [51]. These filters are convolved with a stride of one, so every pixel in the image is utilized. Also, the depth of the VGG-Net is increased with three variants of VGG containing eleven, 16, and 19 layers, where the last three fully connected layers are the same [51]. These fully connected layers contain 4096, 4096, and 1000 neurons, with the output layer neurons corresponding to the number of classes in the classification task [51].

From the proposed architectures, mainly VGG16 and VGG19 are used, with the number referring to the depth of the network. These two architectures are shown in Table 2. The number of kernels starts at 64 in the first convolution layer block and doubles in

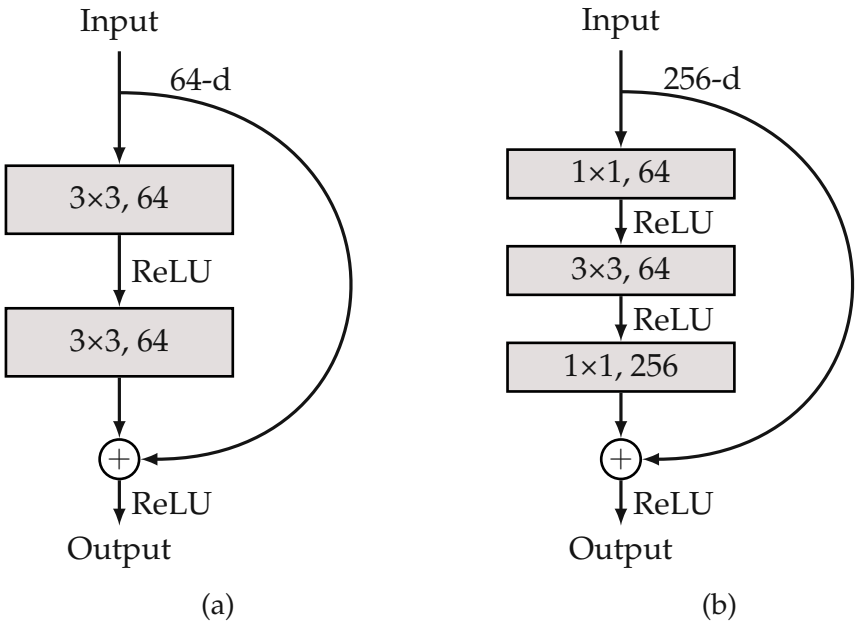


Figure 16: The original residual block (a) contains two convolution layers and a skip-connection. The bottleneck block (b) consists of three convolution layers, where two are 1×1 convolutions to change the dimensions. [17]

every following block until the maximum number of 512 kernels is achieved [51]. After each block of convolution layers, a max pooling layer with a size of 2×2 and a stride value of two is applied [51]. The resulting VGG16 network contains approximately 138 million parameters, which can be modified in training.

ResNet

The network depth of CNNs significantly influences the performance of models, with deeper networks achieving higher accuracies. However, with increasing depth, the network becomes more challenging to train due to the gradient vanishing problem [17]. This is counterintuitive, as deeper networks include the solution space of shallower networks and should perform at least equally well, but they are not able to find this result [17]. He et al. [17] have solved this problem by introducing residual learning with shortcut connections, also referred to as skip connections [17, 54]. These skip connections map the identity and feed the output back after skipping layers, as shown in Figure 16.

In residual learning, the stack of skipped layers is targeted to learn a complex function minus the identity, as it is added through the skip connection [17]. The bottleneck design, shown in Figure 16b, is a further development of the residual block (Figure 16a) and is used for very deep residual networks (ResNets). It uses a stack of three layers compared to the previous two layers in the residual block and a special form of convolutions to transform the dimensions [54]. Recalling the bottleneck block shown in Figure 16b, the input dimension is $width \times height \times 256$. As the identity mapping contains the dimensions, the output of the convolutional layers is required to match the input dimension. The convolutional layers are computationally expensive, and therefore, lower dimensional inputs are preferred. An exception to this is the 1×1 convolution, which is very fast and maintains the $width \times height$ dimension. The depth of the output can be adapted by the number of kernels, which can either result in a dimension reduction or increase. This is used in the bottleneck block as follows.

The first layer in the bottleneck block uses 1×1 convolutions with 64 kernels to reduce the depth before applying the computationally expensive 3×3 convolution layer [17]. The third 1×1 convolution layer uses 256 kernels, which increases the dimensionality to the previous level and matches the dimensions of the skip connection [17].

Another advantage of residual learning is that the skip connections do not increase complexity, and no additional parameters are required [17, 54]. In Table 3 the different ResNets are visualized, and the building blocks can be observed. The input is an RGB image of size $224 \times 224 \times 3$. The first 7×7 convolution and the following 3×3 max pooling with stride two are the same for all variants. Each of the five convolution blocks consists of either regular blocks for the 18 and 34-layer networks or bottleneck blocks for the deeper models. The number of blocks in each convolution block is noted after the brackets. The output size is halved as the network transitions from one convolution block to the next. This is achieved by setting the stride parameter in the first convolution layer in the convolution block to two [17]. When creating the skip connection, the dimensionality is not constant in this case, which requires an adaption to

Table 3: The ResNet architectures for 34, 50, and 152 layers are shown. Whereas the 34-layer architecture uses the original residual block, the other two architectures utilize the bottleneck blocks. (adapted from [17])

34-layer	50-layer	152-layer
Input (224×224×3)		
7×7 conv-64		
3×3 max pool		
$\begin{bmatrix}3\times3\text{ conv-64}\\3\times3\text{ conv-64}\end{bmatrix}\times3$	$\begin{bmatrix}1\times1\text{ conv-64}\\3\times3\text{ conv-64}\\1\times1\text{ conv-256}\end{bmatrix}\times3$	$\begin{bmatrix}1\times1\text{ conv-64}\\3\times3\text{ conv-64}\\1\times1\text{ conv-256}\end{bmatrix}\times3$
$\begin{bmatrix}3\times3\text{ conv-128}\\3\times3\text{ conv-128}\end{bmatrix}\times4$	$\begin{bmatrix}1\times1\text{ conv-128}\\3\times3\text{ conv-128}\\1\times1\text{ conv-512}\end{bmatrix}\times4$	$\begin{bmatrix}1\times1\text{ conv-128}\\3\times3\text{ conv-128}\\1\times1\text{ conv-512}\end{bmatrix}\times8$
$\begin{bmatrix}3\times3\text{ conv-256}\\3\times3\text{ conv-256}\end{bmatrix}\times6$	$\begin{bmatrix}1\times1\text{ conv-256}\\3\times3\text{ conv-256}\\1\times1\text{ conv-1024}\end{bmatrix}\times6$	$\begin{bmatrix}1\times1\text{ conv-256}\\3\times3\text{ conv-256}\\1\times1\text{ conv-1024}\end{bmatrix}\times36$
$\begin{bmatrix}3\times3\text{ conv-512}\\3\times3\text{ conv-512}\end{bmatrix}\times3$	$\begin{bmatrix}1\times1\text{ conv-512}\\3\times3\text{ conv-512}\\1\times1\text{ conv-2048}\end{bmatrix}\times3$	$\begin{bmatrix}1\times1\text{ conv-512}\\3\times3\text{ conv-512}\\1\times1\text{ conv-2048}\end{bmatrix}\times3$
average pool		
FC-1000 (softmax)		

increase dimensionality. He et al. [17] reason that projections are best used for these specific connections. The ResNets are finalized by an average pooling layer and a fully connected layer with 1000 neurons using a softmax activation function to represent the 1000 classes for the ImageNet data [17].

3.7 Training of Neural Networks

With DL models and architectures of CNNs explained, the learning and training of ML models can be detailed. In the following, an overview of important methods is given, focusing on the methods applied in the use case of this thesis.

Mainly, two concepts are considered when training a ML model: supervised and unsupervised learning [12, 54]. Supervised learning is the most straightforward, as each

example data contains a label that specifies the desired output [12, 54]. During training, this label is compared to the output of the ML model, and a loss with a predefined loss function is calculated. Then, the weights of the network are adapted to minimize the loss function and, therefore, approximate the output value to the label. In unsupervised learning, datasets are not provided with labels, and their characteristics need to be determined [12, 54]. Typical tasks in unsupervised learning are pattern recognition, image segmentation, and clustering. However, as this thesis applies supervised learning to classify images, only this concept is considered in the following.

Training a supervised model requires an algorithm that adapts the weights of the network to obtain a better-fitting model for each iteration in the training procedure. This translates to an optimization problem with the main objective of minimizing a loss function [54]. There are different approaches to this optimization problem, which are stated below.

3.7.1 Optimization

To minimize the loss of a network, the weights in the network have to be adapted such that the model better approximates the desired output [12]. To achieve this, the gradient of the loss with respect to each weight is calculated, and the weights are changed in the individual direction [12]. Gradient descent uses the calculated gradient from the whole dataset and updates the weights at once. Because this requires large memory capacities and updates only after processing the whole dataset, other optimizers have been developed. The Stochastic Gradient Descent (SGD) is a commonly used optimizer that only uses one sample to calculate the gradients and updates the weights [54]. This allows for faster computation, but as one sample is often not representative of the whole data, the optimization is unsteady [12, 54]. An adaptation of SGD is the mini-batch SGD, which utilizes a set of samples and averages the adaptations

before updating the weights [54, 12]. Commonly, the mini-batch SGD is used without mentioning the mini-batches. To update the weights

$$\boldsymbol{w_{t+i}} = \boldsymbol{w_t} - \alpha_t \boldsymbol{g_t}, \tag{13}$$

the learning rate parameter α_t specifies what proportion of the gradient $\boldsymbol{g_t}$ is used [54]. This learning rate is a sensitive parameter that needs to be carefully chosen, as a high learning rate might overcompensate. However, a low learning rate prevents learning at all because the gradients are scaled to nearly zero [54].

Another problem with SGD is that it can get stuck in a local minima. Therefore, the momentum, also termed velocity $\boldsymbol{v_t}$ is introduced from an exponentially decaying rolling average of the gradients, which allows for faster training [54, 12]. The updated momentum

$$\boldsymbol{v_{t+i}} = \rho \boldsymbol{v_t} + \boldsymbol{g_t}, \tag{14}$$

is calculated from the velocity $\boldsymbol{v_t}$ scaled with the momentum parameter ρ and the gradient $\boldsymbol{g_t}$ [54]. The updated weights

$$\boldsymbol{w_{t+i}} = \boldsymbol{w_t} - \alpha_t \boldsymbol{v_t} \tag{15}$$

are then calculated analogous but with the scaled velocity $\boldsymbol{v_t}$ [54]. Current implementations, such as the tensorflow framework [2], specify the optimizer SGD and work either with or without the momentum parameter.

Another optimizer is Adaptive Gradient (AdaGrad) [8], which scales the gradient individually by dividing by the square root of the summed squared gradients, and it performs best on sparse gradients [31, 54]. It adapts the learning rate individually for every gradient, which enables faster learning. Tieleman and Hinton [56] have proposed the RMSProp optimizer, which additionally addresses the problem where gradients are scaled to nearly zero by the squared sum in the denominator by scaling it with a moving average.

However, the most commonly used optimizer in conjunction with CNNs is the Adam optimizer [31]. It combines the concepts from AdaGrad, RMSProp, and momentum to form a single optimizer [54]. This is also the optimizer applied in the use case.

3.7.2 Transfer learning

Training of ML models requires extremely large datasets to adapt to the characteristics. However, in many use cases, only a small number of samples is available, which reduces the achievable accuracies of ML models and prevents the possibility of applying complex models to a use case. To solve this problem, Transfer Learning (TL) can be used to train a model on a pre-existing, task-unrelated dataset and then refine it on another task-specific dataset [41, 26, 10]. The main concept is to train a model on a similar domain with ample data before transferring it to the target domain [10].

TL can be used for different forms of knowledge transfer, but with CNN models, usually, only a part of the network is re-trained with the domain-specific data [10]. As the convolutional layers act as feature extractors, these features, especially the low-level features in the first layers, are likely very similar between the domains, and the weights of these layers can be preserved, often referred to as the freezing of layers [10, 26]. Most commonly, only the fully connected layers at the end of the network are re-trained, as they specifically pre-train to the domain of the data [10].

In image classification, a large dataset could contain, for example, 1000 classes of objects, while in the specific application, only the detection of dogs and cats is required. To use TL in this case, the network is pre-trained on the large dataset, and then the last layer consisting of 1000 neurons is replaced by a layer with two neurons. Then, the layers except the last fully connected ones are frozen, and the model is re-trained on the target dataset containing samples of dogs and cats. TL is often used in image classification applications because it improves the performance of large ML models with a small number of target-specific training samples [10]. As many of the CNN architec-

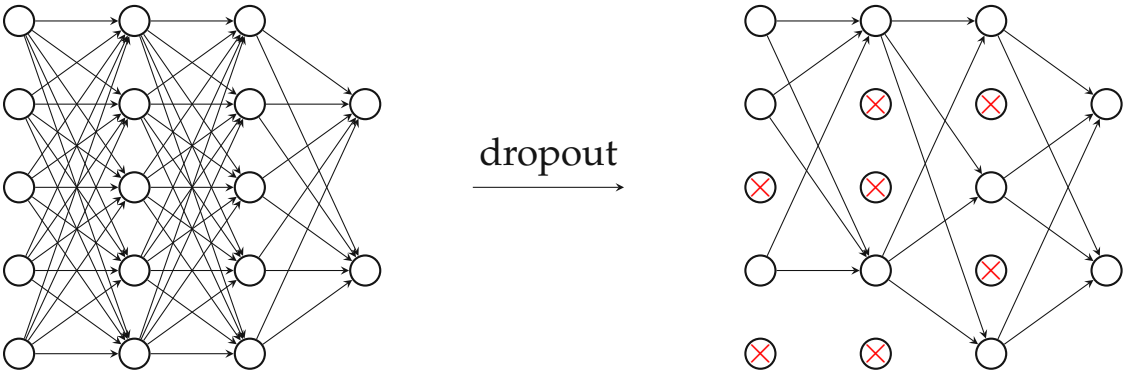


Figure 17: Dropout cuts some connections between neurons during training, which aids the generalization process. The dropout rate in this case is 0.5, which specifies the share of unconnected neurons in a layer. (retrieved from [47])

tures have been developed for the ILSVRC, the ImageNet dataset is often the basis for a TL approach [26].

3.7.3 Overfitting and Dropout

As deep NNs contain many layers of neurons, they can represent complex concepts. However, a common problem in ML is that the model adapts highly to the individual data points during training but loses its generalization capability [54, 12]. This is referred to as overfitting and describes a model with very low training loss but high loss and bad performance on new, unseen data [54, 12].

To counter this, the dropout method has been introduced [52]. When applying dropout to a NN, some neurons and connections are randomly deactivated during training, which results in the model becoming better generalized to the data. The dropout rate specifies the percentage of dropped neurons, which are chosen randomly for each batch in the training [54]. The dropping of specific neurons in the network is shown in Figure 17. Because the network trains on different configurations, the neurons' weights compensate for the missing other neurons, which reduces the chance of neurons specializing to single data points [54]. This is only applied during training, and the output is compensated for the additional connections.

4 Literature Analysis

This chapter summarizes the findings from the literature analysis, which has been conducted in the previous project work [44]. The structured literature analysis has been carried out with four carefully engineered search strings as input for the Google Scholar search [44]. The obtained publications focus on classification and segmentation tasks, primarily in the infrastructure sector, with applications such as crack detection in concrete surfaces or bridges and monitoring of structural integrity of diverse objects [44]. As the classification task is relevant to this thesis, only the publications focusing on this topic are discussed in the following. These publications contribute to the knowledge base in the DS research.

4.1 Results from the Literature Analysis

The publication by Müller et al. [37] proposes a method to classify failed and unfailed material testing specimens from image data [44]. The model uses a subset of the Haralick features obtained from the GLCM, which are then fed into a MLP. They apply this model to three material tests: the uniaxial tension, a notched tension, and an axisymmetric V-bending test [37, 44].

Vijayan et al. [60] have analyzed different methods to detect surface cracks in concrete structures, as manual inspection of these surfaces is often undesirable due to cost and accuracy reasons. They focus on classic image processing methods, as well as ML and DL methods, stating that all methods can be utilized for an automated detection system

[60]. When considering DL methods, CNNs may be best suited for image classification tasks, and the AlexNet architecture seems to be one of the best starting points for the development, as it is very commonly used [60]. Kim and Cho [30] have also used the AlexNet architecture, as they have developed a classification system for concrete cracks captured by a moving drone. With a pre-trained model from MATLAB, they have applied TL to the model and have achieved good results with this architecture. However, they have identified problems with cracks close to the edge of the image [30]. The same MATLAB model has been applied by Rajadurai and Kang [46] in their use case. They have re-trained the last three layers of the AlexNet, with the last layer being adapted to the binary classification task of distinguishing between the presence and absence of cracks in the image [46].

Besides the AlexNet architecture, also the VGG16 architecture has often been proposed in the analyzed literature [44]. Ma et al. [34] have developed a CNN based on VGG16, which has then been adapted and trained using TL. The resulting automated crack detection system for pavement cracks has achieved high accuracies and is capable of real-time application [34]. Huang et al. [23] have applied the VGG16 architecture to classify cracks in different media. However, they have not applied a CNN to a camera image, but on an image resulting from an acoustic emission time-frequency diagram [23]. Because their dataset of the acoustic emission time-frequency diagrams has been too small for sufficient training, they have utilized a pre-trained VGG16 model. This model is capable of identifying edges, stripes, and colors, and then only the last two layers have been re-trained on their dataset [23].

Summarizing the literature analysis, different methods can be applied when developing an automated crack detection system [44]. The publications mainly focus on the detection of cracks in concrete and infrastructure, which is not the topic of this thesis, but as the failure of specimens shares characteristics with those cracks, the available literature should contain relatable insights [44]. Also, different approaches in detecting cracks from image data have been identified, ranging from simple image processing,

over the extraction of textural features in combination with a ML classifier, to sophisticated CNNs [44].

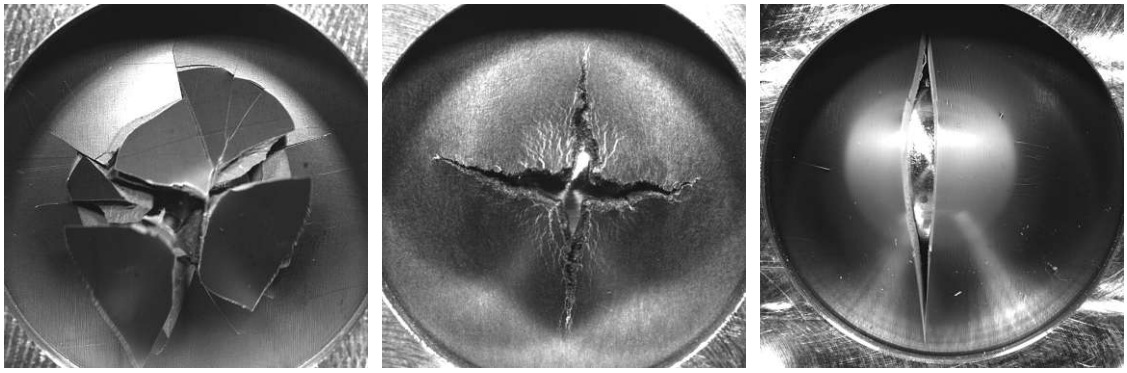
5 Exploratory Data Analysis

This chapter gives a brief introduction to the dataset to allow for an understanding of the data available to this use case. This exploratory data analysis has also been part of the project work [44], and therefore, only the most important insights are presented in the following.

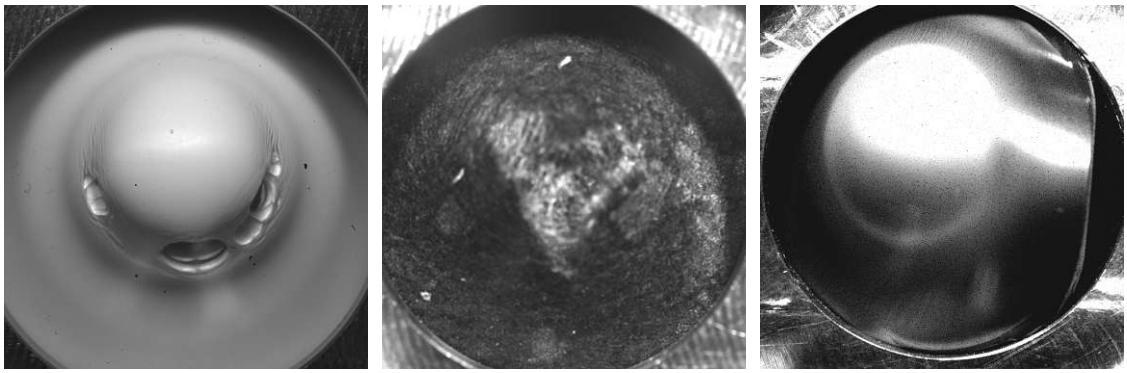
5.1 Introduction to the provided data

The dataset comprises 963 individual dynamic puncture tests featuring 82 different materials [44]. The testing temperature of the specimen varies with a minimum of $-40\,°C$ and a maximum of $90\,°C$ [44]. The data for each test include a clipped image series captured by a high-speed camera. This series is not of constant length and consists of 51 to 116 images. Also, a CSV file is available, where essential parameters, such as various testing conditions and test settings, are listed.

The two most important parameters are the Failure Frame, which is the label for this thesis, and the Failure Case. The Failure Frame specifies the frame where the first visible failure occurs [44]. As the term failure can be used for many states, the definitions described in ISO 6603-1:2000 [27] are applied. The term "failure" refers to a visible break in the surface of a specimen, whereas a "crack" refers to any observable fissure that does not extend through the entire thickness of the specimen [27]. The crack fully extending through the specimen is referred to as a "break" [27]. The "shattering" term is used when the specimen breaks into multiple pieces after the impact [27].



(a) Case 1 (b) Case 2 (c) Case 3



(d) Case 4 (e) Case 5 (f) Case 6

Figure 18: The six different Failure Cases in the provided data by 4a engineering GmbH show different failure behaviors. [44]

Six Failure Cases have been introduced, as the tests within the dataset show drastically different failure behavior, and the differentiation between the cases could allow for more targetted ML models [44]. This could be achieved by training specific models to the different Failure Cases, which could result in better prediction results. Exemplary images of the six Failure Cases are shown in Figure 18. Failure Case 1 in Figure 18a shows the shattering of a brittle specimen into many fragments along a circular break. The second Failure Case in Figure 18b is characterized by small deformations before an oriented break occurs, which marks the point of failure [44]. Also dependent on the orientation, Figure 18c shows a ductile material with the formation of a single crack. In Failure Case 4, as visualized in Figure 18d, the specimen experiences large deformations prior to the failure. The failure then occurs when local neckings form around a circular area [44]. Experiments of this failure behavior, especially with a shiny surface, can reflect under the deformation, resulting in an overexposed area in the image. The

fifth Failure Case is assigned to experiments with no assignable case, and one example is shown in Figure 18e. Failure Case 6 refers to experiments that fail at the specimen clamping of the supporting structure, as can be seen in Figure 18f [44].

Experiments of Failure Cases five and six are considered invalid experiments and have been excluded from the dataset. Overall, 957 valid experiments have been obtained, which are used to develop the different approaches [44].

5.2 Results of the Data Analysis

Summarizing the data analysis results, three factors influence the development of an automatic evaluation model. The introduction of Failure Cases could prove beneficial when specific models are applied [44]. However, the number of experiments in the dataset is too low to distinguish between the Failure Cases. As ML models require training, validation, and test data, the remaining experiments for determining the performance of a model reduce drastically, lowering the confidence in the results [44]. The second factor relevant to the implementation is the varying number of image frames within an experiment, contributing to the complexity of the approaches [44]. Lastly, the provided dataset contains experiments with different failure behavior and materials. Detecting the Failure Frame manually can be very difficult, as some specimens fail within a range of several images, and the exact frame is not apparent [44]. However, other specimens clearly fail from one frame to the next, and the Failure Frame can be detected easily. As identification of the Failure Frame varies in ease, label accuracy differs accordingly [44].

6 Used Approaches

In this chapter, the three approaches that have been used in this thesis are derived. This marks the beginning of the design and development process within the DSRM, where the artifact is created. Drawing from the literature and the data analysis, essential conclusions can be made, leading to these approaches.

The image series provided vary in length, the tested material, and their failure behavior. Therefore, in the first step, the problem of Failure Frame detection translates to a classification task for individual images into unfailed and failed specimens. This classification is achieved differently for each of the three approaches. After that, with the individual predictions for all images of an experiment, the point in the series has to be defined, where the state change happens, translating to a segmentation task. In this secondary step, changepoint detection, as described in Section 3.4, is used to identify the frame where the failure occurs from the series of classifications.

The first, simple image-processing approach processes the images, and a series of output values are calculated without using a ML algorithm. The information captured in the images changes over time as the specimen is hit with the impactor and signs of failure start to form. When detecting the Failure Frame manually by visual inspection, neighboring images are compared in the series, and changes can be identified. The image with the first visible failure shows significant local changes in the pixel values compared to the image before. Therefore, the main focus of this approach is to mimic manual detection by analyzing the differences between images and determining the Failure Frame from this information.

A second approach originates from the literature. As described by Müller et al. [37], a system using Haralick features in combination with a NN achieves promising results. Their use case is also centered around material tests, but they investigate other testing procedures and have a different image generation method. Because the model developed has achieved high performance, these features and a downstream NN have also been selected and implemented in this thesis.

The third approach classifies the images using CNNs. Most of the publications found during the literature search suggest the use of these networks for classification tasks, mainly when applied to crack detection or the identification of defects in images. Therefore, CNNs has also been chosen as an approach for this thesis.

Summarizing, the three approaches to the classification tasks are:

- Simple Image Processing
- Haralick Features with a NN
- Models using CNNs

7 Implementation

In this chapter, the implementation of the use case is described, which corresponds to the demonstration process in the DSRM. The Python programming language is utilized, and for the NNs, the tensorflow framework [2] is used. Several additional packages, mainly numpy [16], pandas [55], ruptures [57], mahotas [7], and PIL [6] are also used. For visualizations, the matplotlib [24] and seaborn [61] packages are employed.

As the findings from the literature suggest, three different approaches as described in Chapter 6 are chosen and implemented. A pipeline containing critical steps in the procedures is developed to enhance the consistency in implementing the three approaches, as detailed in Section 7.1. The individual characteristics of the three approaches are described in separate sections, starting with the simple image processing approach in Section 7.2, continuing with the approach using Haralick features and a NN in Section 7.3, and lastly stating the details of the approach that uses CNNs in Section 7.4.

7.1 Pipeline

This section details the general pipeline for detecting the Failure Frame and is shown in Figure 19. All approaches start with the *Image Series* of the puncture test. A *Pre-Processing* step is applied to prepare the images. Then, for approaches using a ML model, a *Training* step returns a model to predict the states of images in the *Predicting* step. However, the simple image processing approach directly uses the *Processing* step

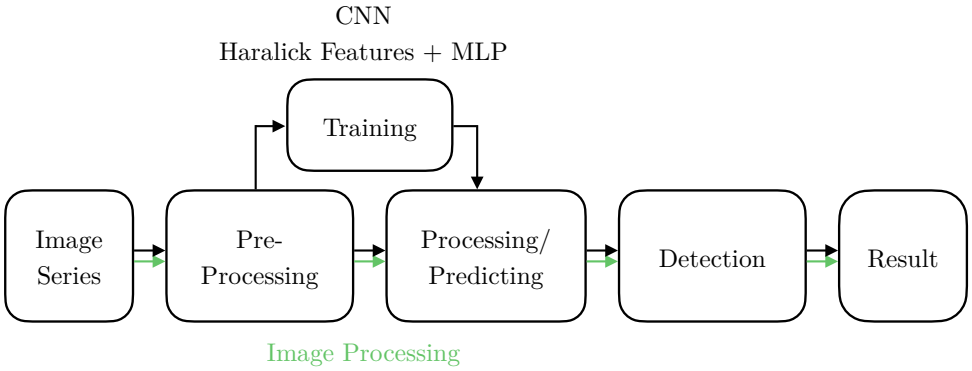


Figure 19: The general pipeline builds the foundation for developing every approach. Whereas the simple image processing approach uses no ML model and can directly be used to obtain a result, the other approaches (Haralick features with a NN and CNNs) require a *Training* step.

without the need to train a model. The pipeline is then continued by the *Detection* of the Failure Frame, and, ultimately, the *Result* is obtained. These steps are described in the following.

7.1.1 Image Series

The implementation is started by obtaining the experiment data consisting of a directory containing a CSV file and a folder with the image series. This image series is used to detect the Failure Frame. The CSV file contains specific data for the individual experiments, such as the material type, testing temperature, and Failure Frame. The Failure Frame entry contains the manually labeled Failure Frame number and is used to train an ML model or evaluate the prediction's performance.

7.1.2 Pre-Processing

The data is manipulated and prepared for the next steps in the *Pre-Processing* step. Approaches two (Haralick Features) and three (CNNs), using a ML model, require different datasets to allow for the model's independent training, validation, and testing. Therefore, the experiments are split into three datasets by applying the following logic in the first step.

All experiments are read into a pandas dataframe containing the properties of the CSV file. As the failure behavior is highly dependent on the material, the material type is used as a key to split the experiments. By iterating over the list of unique materials, the experiment numbers are distributed to the three datasets in each step. 70% are assigned to the training, and 15% each to the validation and testing dataset. This works for materials with many experiments, but a regulated splitting procedure is used for materials with five or fewer experiments. The experiments are distributed to the training and testing dataset for materials with two experiments, and a copy of the training experiment is assigned to the validation dataset. This ensures that the material is represented in every dataset, but the examples of the testing dataset have not been seen by the ML model in the training procedure. With the material having three experiments, the experiments are split evenly. When having four experiments, the additional one is assigned to the training set, and in the case of five experiments, the last one is added to the testing set. These regulated splits for materials result in a shift of percentages in the actual splits, with the training dataset containing 62.6% (599 experiments), the validation dataset containing 18.8% (180 experiments), and the testing dataset containing 18.7% (179 experiments). The sum of all experiments in these datasets is one experiment larger than the original number of experiments, as one experiment is in the training and validation dataset.

The other *Pre-Processing* steps, including the label generation for the approaches using ML models, vary and are explained in more detail in the individual sections. The pre-processed output is then either used to train a ML model or directly fed to the *Processing* or *Predicting* step.

7.1.3 Model Training

The training of a model only applies to the approaches using ML models. To train a model, the training and validation datasets from the *Pre-precessing* step are used. The testing dataset is only used after the ML model has been trained completely. The pre-

viously determined Failure Frames are used to generate labels for each image. Because the number of images is inconsistent, every image is labeled individually, with all images before the specified Failure Frame labeled as zero. The image at and all images after the Failure Frame are labeled as one. Every approach is based on this labeling and, therefore, also returns a prediction for every processed frame.

Each approach uses different ML networks described in the respective sections, but all are implemented with the tensorflow framework [2]. Also, the training of the networks is the same for each model, as they use the Adam optimizer and the BinaryCrossentropy loss function, both with the default settings defined in the tensorflow package [2].

7.1.4 Processing/Predicting

In the *Processing* or *Predicting* step, the prepared data is manipulated to obtain an output. This differs between the approaches because the simple image processing approach does not use a ML model and calculates the output directly. The other approaches, however, use a trained model which predicts a result for the given input data. The output is a list containing a value between zero and one for each processed frame in the image series, with zero corresponding to no failure and one to an image with a failed specimen.

7.1.5 Detection

The output needs to be evaluated to return a specific Failure Frame. Therefore, the obtained series of numbers is processed via changepoint detection, described in Section 3.4. The changepoint detection uses the Python package ruptures [57] and the therein implemented method *Dynp*.

The algorithm takes a list of the classification predictions and returns the detected changepoint as an integer value that specifies the index in the processed predictions

list. The least squares deviation cost function (CostL2) is specified as the model because this function detects shifts in the mean of signals [57]. Then, the predictions are fitted to the changepoint model and evaluated to obtain a prediction for the changepoint. As the specimen transitions only once at the Failure Frame from an un-failed to a failed state, there is only one changepoint to predict. This predicted changepoint is the index in the image series, which is then used to identify the Failure Frame number by extracting the number from the corresponding image name.

7.1.6 Results

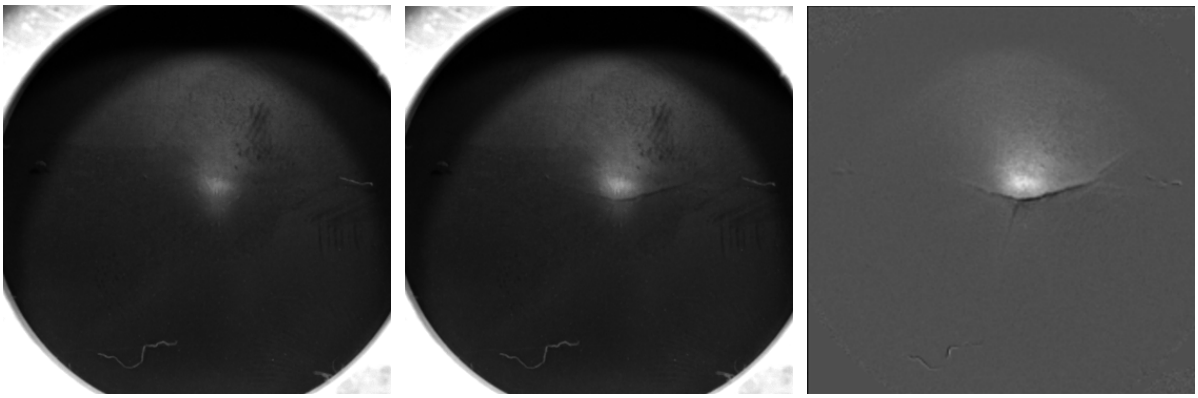
In the last step of the pipeline, the Failure Frame number from the detection step is retrieved and either written to a file or further processed for evaluation and visualization purposes. This marks the pipeline's ending point, which is the same for all approaches. However, in the following, the individual approaches are described.

7.2 Approach 1: Simple Image Processing

The first approach is referred to as simple image processing. Due to the time dependence captured by the high-speed images, an intuitive frame differencing as detailed in Section 3.1 is utilized in this approach. It will focus on the changes between two consecutive image frames by subtracting individual pixel values. This approach is straightforward but acts as a benchmark for the more sophisticated approaches.

7.2.1 Implementation of the Simple Image Processing

The images are processed by iterating over the available images and subtracting the pixel values of the previous image from the index image. The resulting difference matrix $\mathbf{D}$ contains the changes between the two frames $\mathbf{I}_i$ and $\mathbf{I}_{i-1}$ and is obtained according to Equation (7).



(a) (b) (c)

Figure 20: The image one frame before the Failure Frame (a) shows no failure and contains imperfections and the specimen clamping. The image at the Failure Frame (b) also contains these imperfections, the specimen clamping, and the failure in the center. The difference image (c) results from the image differencing method applied to those two consecutive images. The failure is highlighted, and other texture elements, including the specimen clamping and surface imperfections, are removed.

For illustration purposes, two exemplary images of an experiment are depicted in Figure 20, showing one image one frame before the failure (Figure 20a), and one at the Failure Frame (Figure 20b). While the failure is not formed in the first image, it is visible in the second image. Due to the first appearance of the failure in the second image, this is the Failure Frame. It also has to be noted that both images contain the edges of the specimen clamping and some unidentifiable imperfections that might stem from dust or other contamination. These imperfections are not a relevant factor in determining the Failure Frame and should not influence the image classification process. With the simple image differencing method applied, the pixel values of the two images are subtracted, resulting in an image shown in Figure 20c.

The difference frame captures the crack, which can be seen in more detail as the other information in the frames is not changing significantly. The ring of the specimen clamping is almost not visible, and the other imperfections are noticeably less prominent. Therefore, in this case, the differencing method highlights relevant information while reducing other static components in the images.

To condense this information into a single value for one difference frame, the spread

$$s = \max(\mathbf{D}) - \min(\mathbf{D}) \tag{16}$$

is calculated by subtracting the minimum value from the maximum value of the matrix $\mathbf{D}$.

Advancing with the index in the image series returns a spread per image pair. These spreads are then added to a list of spreads for further processing. As the spreads always require two images for one value, the first image in the series has no spread value, and the list of spreads is one item smaller than the number of images in the original series. To maintain the length of the series, the calculated first spread value between the first and second image is also used as the starting point for the first value of the series, even though there is no spread for the first image. This allows for the same indexing as in the original series without having a drastic change in the spread values at the start.

After the spread values for the experiment are obtained, the values are normalized to values between zero and one. This normalized list is then passed to the changepoint detection function, where the Failure Frame is calculated as described in Section 7.1.5.

7.3 Approach 2: Haralick Features and Neural Network

The second approach uses Haralick features with a simple feed-forward NN. The Haralick features as described in Section 3.2 have been utilized by Müller et al. [37] to develop an automated ML fracture detection system for material testing.

7.3.1 Implementation of the Haralick Features and Neural Network

The Haralick features are calculated using the mahotas package [7] available for Python. Using this package, the 14 Haralick features can be obtained for each of the four directions, adding up to 56 individual features. These features are then used as a represen-

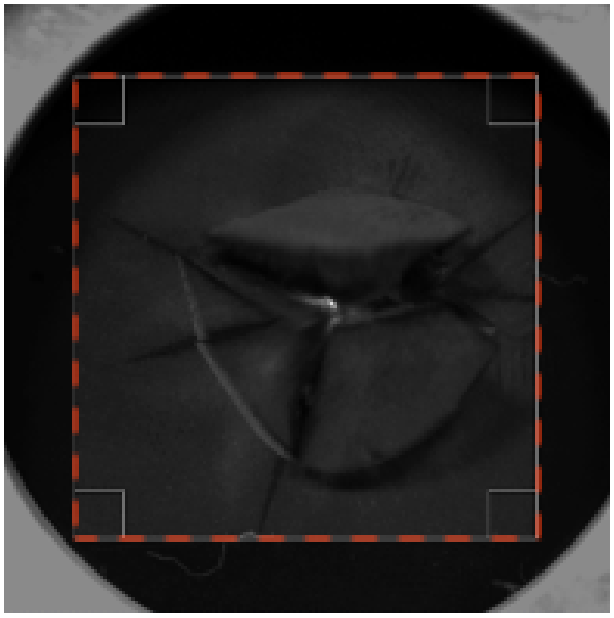


Figure 21: The Haralick features are extracted from a cropped image without the specimen clamping, as only the pixels within the red square are used.

tation of the image. During the implementation phase, very low correlations between the Haralick features and the image label are encountered, which results in an unusable ML model. As the Haralick features are focused on texture, the edges from the specimen clamping, which are visible in each image, can largely influence the features and reduce the discriminatory power. Therefore, the images are center cropped to 480×480 pixels to only contain the specimen without the specimen clamping, as shown in Figure 21. This increases the correlation of the features to the label and allowed for a ML model to be trained with success.

The sequential ML model is built within the tensorflow framework [2] and consists of seven dense layers and one dropout layer, which is located before the last dense layer. The network architecture is developed by trial and error. The dense layers, except the last one, all use the ReLU activation function and contain 64, 128, 2048, 4096, 2048, and 64 neurons.

The last dense layer consists of one neuron, uses the sigmoid activation function, and returns values between zero and one as the ultimate output of the ML model. These values can be interpreted as the probability of a failure in the corresponding image.

The training of this network uses the testing and validation datasets for a total of 200 training epochs. During this training, the area under the curve (AUC) has been monitored, and only the best model has been saved. The AUC is a quality measurement for binary classifiers. For the implementation, the AUC metric from the tensorflow framework has been used [2].

The predictions for the testing dataset are obtained using the trained model. These predictions are then passed to changepoint detection, which returns a predicted Failure Frame as described in Section 7.1.5.

7.4 Approach 3: Convolutional Neural Networks

The third approach utilizes a CNN to classify each frame into unfailed and failed frames. Three different network architectures are compared within this approach, as well as four different Data Models, which are explained in detail in the following.

7.4.1 Used Convolutional Neural Network Architectures

Three CNN architectures have been used in this approach: AlexNet, VGG16, and ResNet50. The general characteristics of these architectures are stated in Section 3.6. Therefore, only the specific adaptations to fit the application in the use case are detailed below.

AlexNet

The AlexNet network is the only CNN trained solely on the dataset available from the use case and does not use TL. The architecture is oriented on the network proposed by Krizhevsky et al. [32] with five convolutional layers and three dense layers. The AlexNet has been designed to classify images among 1000 classes, with one output neuron per class. As the classification task in this use case is binary, only one neuron is

used in the final layer of the network. The activation function for this single neuron is set to sigmoid to obtain predictions within a range from zero to one.

VGG16

The second network used, VGG16, has been pre-trained on the imagenet[1] dataset and retrieved in a pre-trained state from the Keras library.

VGG16 has also been designed to distinguish between 1000 classes, which requires adaptations for this use case. Therefore, the convolutional and pooling layers are used without changes, and additional layers for the classification are added manually. These layers are two dense layers with 4096 neurons each and ReLU activation, followed by a dropout layer and a final dense layer with one neuron and the sigmoid activation function. The base network, which is the unmodified part of the architecture, is not further trained on the dataset because it is frozen during training. Only the manually added layers are trained in the training step.

ResNet50

ResNet50 is the last architecture implemented and pre-trained on the ImageNet dataset as VGG16. The base model of ResNet50V2 is imported from the Keras library, and analogous to the other architectures, the last dense layer is replaced with a layer containing one neuron and the sigmoid activation function.

7.4.2 Data Models

The CNN architectures require an input image of size $224 \times 224 \times 3$, with the third dimension usually used for different color channels. Since the images in this use case are grayscale and have only one channel, the images must be transformed to fit the

[1] The imagenet dataset has been used for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The full dataset is not publicly available but can be requested under: https://www.image-net.org/index.php.

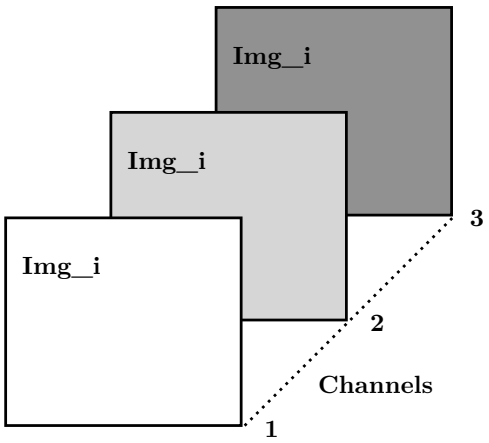


Figure 22: For Data Model 1_Img, the information from the selected image frame is tripled and fed to the three channels of the input.

required three channels. Three times more values are available in the required input image than in one grayscale image, which allows for engineering the input to be more sophisticated. In the following, the four methods of filling the available three channels are referred to as Data Models.

Data Model: 1_Img

This Data Model utilizes one image frame as an input to the model and is also the most straightforward Data Model. The pixel values of the frame are filled into every channel as shown in Figure 22. This Data Model does not provide additional information to the input but transforms the image into a processable format for the CNN. The implementation of this Data Model is noticeably more straightforward, as every image only requires the image itself, without the requirement of referencing and retrieving information from other images in the series.

Data Model: 3_Img

The images within one experiment are connected to another, as the images capture the state of the specimen in a timeline. Recalling the image differencing approach as used in the simple image processing approach, the progression in time allows for the detection of changes and is considered in this Data Model. It uses three images from

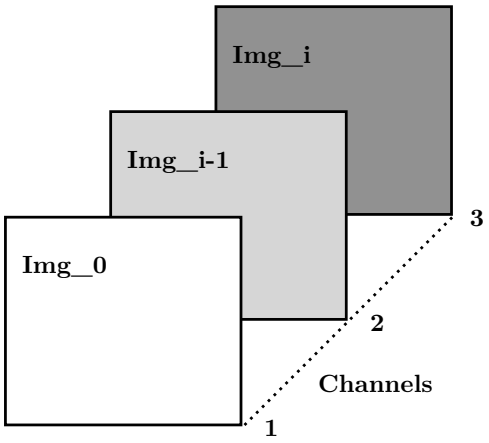


Figure 23: In Data Model 3_Img, the three input channels are filled with the original image, the previous image, and the image at the current index.

the experiment at each timestep. The first channel is filled with the first image in the series, ensuring that the un-failed state of the specimen is represented. This might help the model to detect failures in the index image, as the first image could act as a reference. The second channel contains the image one frame before the actual index, and the third channel is filled with the image at the current index, which can be seen in Figure 23.

Data Model: 4_Img_1

This Data Model mimics the image differencing concept as four consecutive images are extracted from the image series and differenced to form the tensor. Therefore, the first channel is filled with the difference between the index image and the image one step before. The second channel contains the subtraction of the index image and the image two steps before, and ultimately, the third channel is filled with the index image subtracted by the image three steps before. This is depicted in Figure 24 and results in an input containing the changes from the previous three images to the current index image. These changes from images with different indices before the index image to the index image might provide useful information to the model and allow for better classification results.

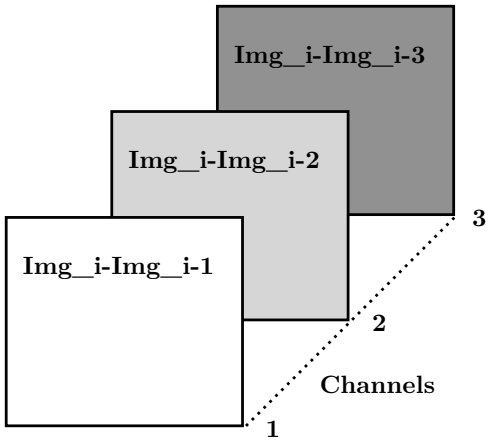


Figure 24: In Data Model 4_Img_1, the three input channels are filled with the differences from 3 consecutive image frames before the index to the index frame.

Data Model: 4_Img_2

Data Model 4_Img_2 uses a mix of the Data Models above, as the first channel contains the index image, the third channel contains the first image in the series, and the second channel is a difference between the index image and the image one step before. Figure 25 shows this Data Model.

Having described the three CNN architectures and the four Data Models, twelve combinations of these architectures and Data Models can be obtained and build the CNN models implemented in this use case. These CNN models will be referred to by both their architecture and Data Model, for instance, VGG16 4_Img_2.

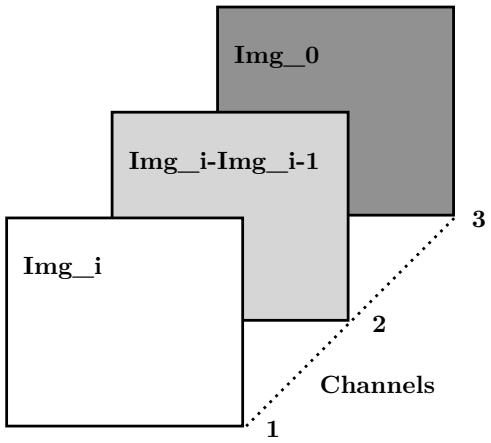


Figure 25: In Data Model 4_Img_2, the three input channels are filled with the original frame, the difference between the index image and the previous frame, and the index image

7.4.3 Implementation of the Convolutional Neural Networks

The implementation for the CNN approach is consistent for the twelve models and only differs in the used network architecture and input tensor. The dataset splits are generated according to the procedure detailed in Section 7.1.2. The training and validation experiments are retrieved from the splits, then transformed into datasets and shuffled. Then, the network architecture is loaded and compiled as stated in Section 7.1.3 before the model is trained. The models are trained using the training and validation data for a maximum of 80 episodes. However, the training is stopped early if no training progress is made, and an automated adjustment to the learning rate is made if necessary. During training, the binary accuracy of the validation dataset is monitored, and only the best-performing model is saved. Noticeable differences in the training times have been observed. However, as the workload of the server at the Institute for Lightweight Design and Structural Biomechanics (ILSB) has not been constant and different amounts of resources have been used during the pieces of training, an exact statistic for the training times is not obtainable. The general tendencies are that AlexNet is the fastest to train by a large margin, and VGG16 and ResNet50 have similar training times, with ResNet50 requiring a little less time per epoch.

The best model is used in the *Predicting* step of the pipeline to predict the labels of the test dataset. Then, the predictions are used as input to the changepoint detection, which returns a predicted Failure Frame for each experiment. These predicted Failure Frames are either stored and written to a file or can be further processed for evaluation purposes.

8 Results

In this chapter, the results of the use case are presented, which evaluates the obtained artifact in the DSRM. The test data used in this chapter differs from the first approach to the other approaches. As the simple image processing approach does not need training, all available data is used as test data, and 957 experiments are tested. All other approaches require dataset splitting, which reduces the number of testing experiments to 179, as stated in Section 7.1.2. The simple image processing approach and the Haralick features approach only contain one model each, whereas twelve models are evaluated from the CNN approach. The output of each model is the predicted Failure Frame. As there is a series of images for the experiments, the accurate predictions and the distribution of the predictions are essential criteria. The image series of the experiments are captured at different framerates, ranging from 10000 to 22500 frames per second. This means that a time difference of around 44µs to 100µs occurs between two frames, which is relevant when evaluating the results. In the following, only the frames are stated, as the time difference can be calculated from the framerate. The used metrics and results for this chapter are the difference between the predicted Failure Frame to the label frame, the frame accuracy, the interval and worst outliers, and the Inference Time. These are stated below, and an additional showcase section and analysis of the worst predictions of the best models is given.

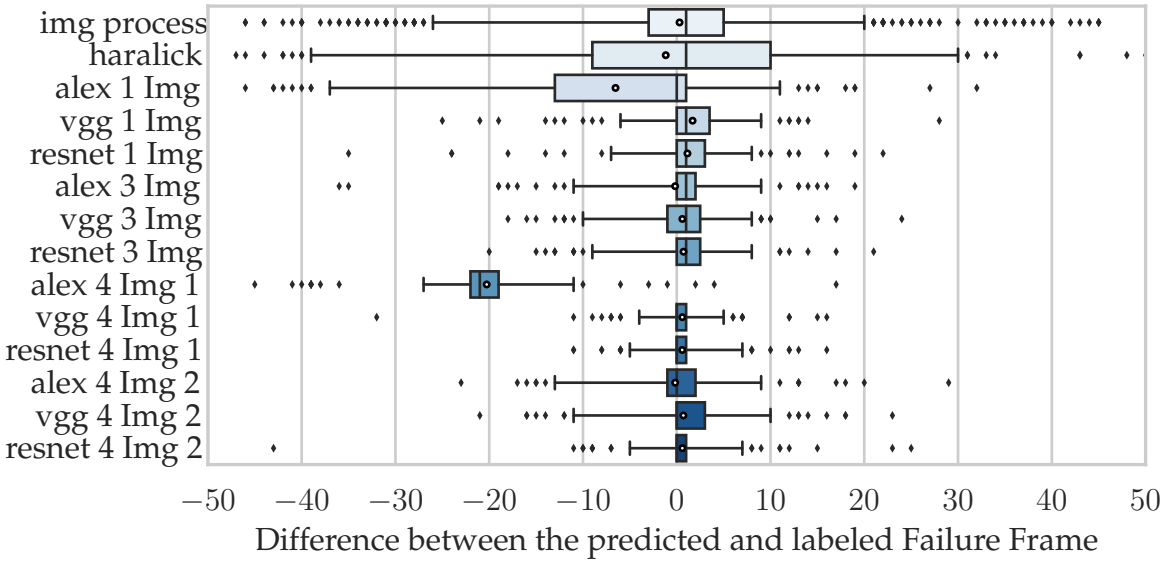


Figure 26: The distribution of differences between the predicted and labeled Failure Frame for each individual model shows different levels of performance. The whiskers of the boxplot expand to the 5th and 95th percentile, and the means are marked as white dots.

8.1 Difference to the Label Frame

This metric is defined as the difference between the predicted Failure Frame and the labeled Failure Frame and is assessed in frames. It indicates how close the predictions are to the desired Failure Frame and is shown in a boxplot in Figure 26. The settings for the box are the default for the interquartile range, spanning from the first quartile (25th percentile) to the third quartile (75th percentile). The median is shown as a vertical line, and the mean difference is marked with white circles, while the whiskers extend to the fifth and 95th percentile (values interpolated and rounded).

The AlexNet 4_Img_1 model shows a significant shift in the mean of the predictions, which results from a poorly trained model. In the training, the model does not adapt to the behavior of the data but generally predicts a value close to 0.5. This results in unusable predictions for the changepoint detection. Upon further analysis, AlexNet 1_Img also shows this problem, which explains the wide distribution as shown in Figure 26.

Besides the models using AlexNet and Data Model 1_Img and 4_Img_1, the other models' mean values are near the targeted zero difference number. It is also noticeable that

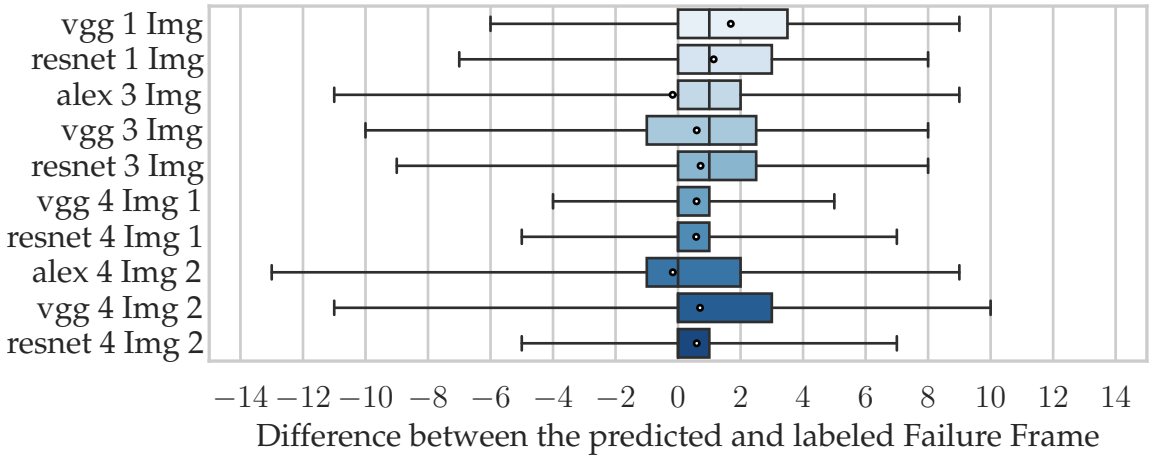


Figure 27: Only the difference results of the better performing models are shown, with the image processing, the Haralick feature, the AlexNet 1_Img, and the AlexNet 4_Img_1 model removed. The outliers are not shown, and the axis limits are adapted.

the predictions from the image processing and Haralick approaches are wider spread than the CNN based approaches, but the simple image processing approach performs better than the Haralick approach. However, with an average number of around 100 frames per experiment, differences in the region of more than 30 frames indicate poor performance. Generally, besides the AlexNet 1_Img and 4_Img_1 models, the models of the CNN approach show better results in the boxplot compared to the simple image processing and Haralick feature approaches. To further discuss the performance of these CNN models, Figure 27 only contains those ten models, excluding AlexNet 1_Img and AlexNet 4_Img_1. The outliers are removed for better visibility, and the whiskers extend analogously to the fifth and 95th percentile. Three models, VGG16 4_Img_1, ResNet50 4_Img_1, and ResNet50 4_Img_2, have the smallest interquartile range located between the differences of zero and one. The whiskers of the VGG16 4_Img_1 model span the shortest distance from minus four to plus five frames, resulting in the best performance in this metric.

8.2 Frame Accuracy

The Frame Accuracy metric measures the percentage of predictions within a specific deviation from the label frame. Opposed to the difference metric stated above, this metric is centered on the labeled frame and focuses on the distribution of a symmetrical interval. This is especially useful since implementing an automated detection system requires a high rate of predictions within a few frames around the actual Failure Frame. In the following, the results for specific Frame Accuracies are stated. As the AlexNet architecture, in combination with Data Models 1_Img and 4_Img_1, returns unusable predictions, these two models are not considered in this metric.

8.2.1 5-Frame Accuracy

The 5-Frame Accuracy of a model is the percentage of predictions located between five images before and five images after the labeled image frame. Figure 28 visualizes the results for the 5-Frame Accuracies of the models. With a 5-Frame Accuracy of 36%, the model using Haralick features performs worst, followed by the simple image processing model with 60%. Therefore, these two models are not considered for the following Frame Accuracies with smaller intervals, as all CNN models in Figure 28 outperform those models.

The CNN models are grouped by the Data Models to outline the influence of the used network architecture. AlexNet performs inferior to the other architectures on the same Data Model, with 5-Frame Accuracies of 78% for 3_Img and 80% for 4_Img_2. For Data Models 1_Img, 3_Img, and 4_Img_2, ResNet50 achieves higher accuracies than VGG16. However, when considering Data Model 4_Img_1, VGG16 shows the highest accuracy of 90%, followed by ResNet50 with 88%.

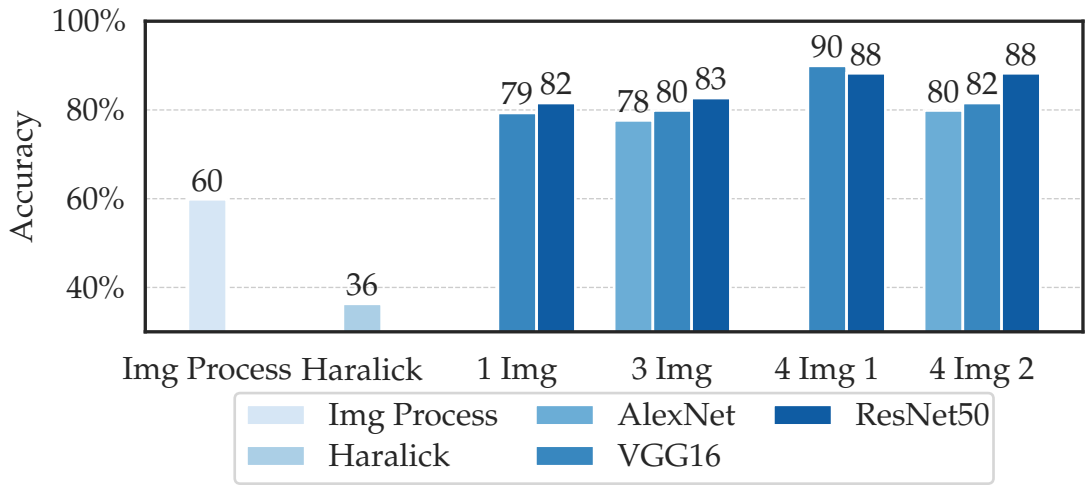


Figure 28: The 5-Frame Accuracy of all models except the AlexNet 1_Img and 4_Img_1 model is shown in percent, with a higher percentage indicating better performance. The CNN models are grouped by the used Data Models with the bar label rounded to an integer percentage.

8.2.2 3-Frame Accuracy

The 3-Frame Accuracy is analogous to the previous metric, with the percentage of predictions being in an interval from three frames before to three frames after the Failure Frame. The 3-Frame Accuracies in Figure 29 exhibit similar behavior as seen in the 5-Frame accuracies, with VGG16 and Data Model 4_Img_1 achieving the best result of 81%. ResNet50 with the same Data Model has a 3-Frame Accuracy of 80.4 % and, in

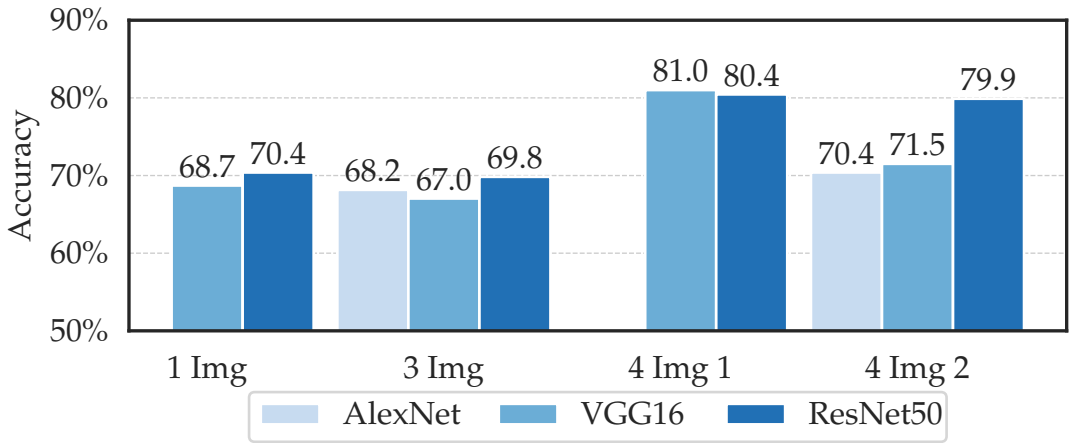


Figure 29: The 3-Frame Accuracies for the considered eleven CNN models are grouped by the Data Models with the bar labels marking the percentage rounded to one decimal. Higher percentages indicate better performance of the model.

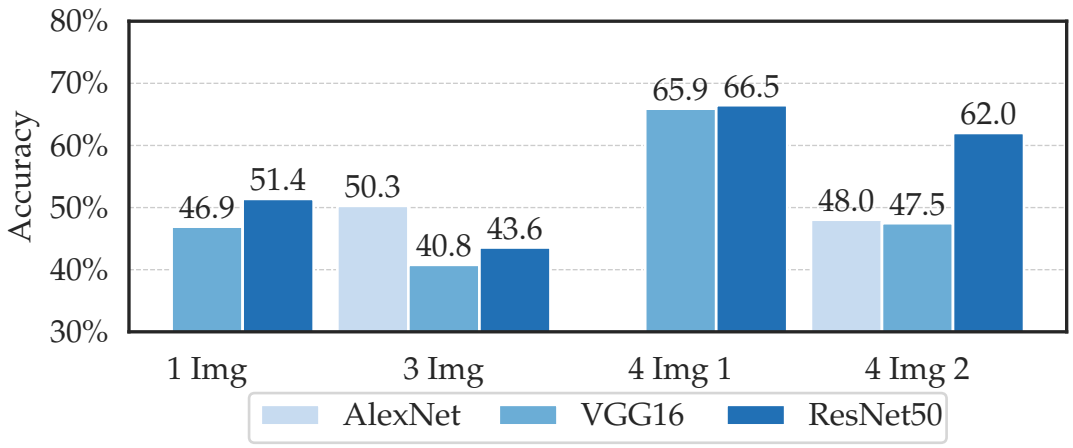


Figure 30: The 1-Frame Accuracies for the considered eleven CNN models are grouped by the Data Models with the bar labels marking the percentage rounded to one decimal. Higher percentages indicate better performance of the model.

combination with Data Model 4_Img_2, an accuracy of 79.9%. While VGG16 has the highest accuracy in combination with one Data Model and medium performance with the other Data Models, ResNet50 achieves similar and high accuracies with two Data Models (4_Img_1 and 4_Img_2).

8.2.3 1-Frame Accuracy

The 1-Frame Accuracy accounts for the percentage of all predictions that are within plus or minus one frame of the Failure Frame, and is shown in Figure 30. Data Model 4_Img_1, in conjunction with VGG16 and ResNet50 achieves the best performance, with the ResNet50 architecture performing slightly better with a 1-Frame Accuracy of 66.5%. However, the VGG16 4_Img_1 model at 65.9% delivers similar performance. ResNet50 with the 4_Img_2 Data Model detects 62% within the interval, but all other models perform considerably worse with 1-Frame Accuracies between 40.8% and 51.4%.

8.2.4 True-Frame Accuracy

The True-Frame Accuracy only considers the percentage of true predicted Failure Frames. This accuracy is an essential indicator of how accurately a model can predict the Fail-

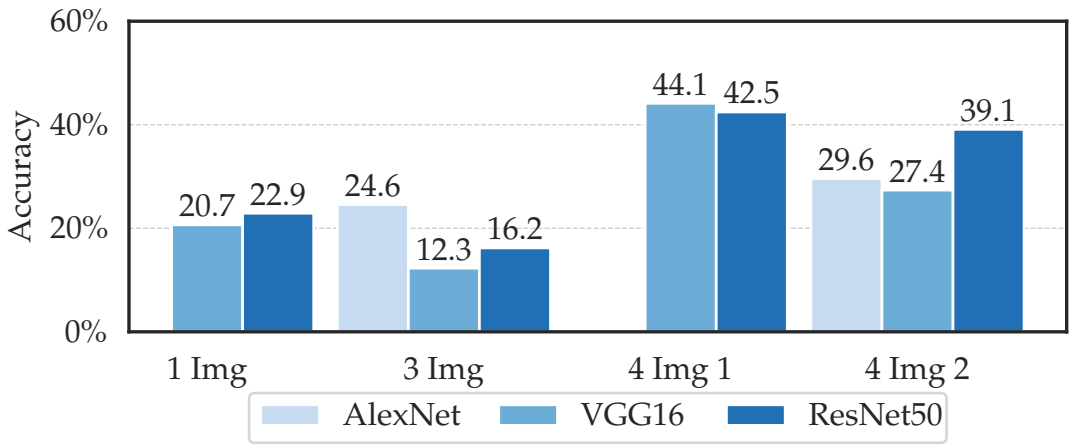


Figure 31: The True-Frame Accuracy for the CNN models are the percentage of true predictions. The bar labels are rounded to percentages with one decimal. Higher percentages indicate better performance of the model.

ure Frame without deviation. The results are visualized in Figure 31. With an accuracy of 44.1%, VGG16 with Data Model 4_Img_1 performs best, followed by ResNet50 and Data Model 4_Img_1 with 42.5% accuracy. ResNet50 also achieves good accuracy in combination with Data Model 4_Img_2 at 39.1%. The other models show low accuracies between 12.3% and 27.4%.

Concluding from the specific Frame Accuracies above, the best model combines Data Model 4_Img_1 and VGG16 followed by Data Model 4_Img_1 and ResNet50. The ResNet50 model has a slightly higher 1-Frame Accuracy but performs inferior to the VGG16 model for other Frame Accuracies, especially the True-Frame Accuracy. Whereas ResNet50 also shows comparable results with Data Model 4_Img_2, VGG16 only achieves high results combined with Data Model 4_Img_1.

8.3 Interval and Worst Outliers

The interval metric focuses on the spread of frames within which a specific percentage of predictions are located. It is calculated by computing percentiles and, after that, retrieving the spread. The

$$90\%\text{-}Interval = frame\_d_{95th} - frame\_d_{5th} \tag{17}$$

uses the fifth and 95th percentile of a model's frame differences $frame\_d$ and then calculates the spread as an integer number of frames. This metric is not centered around the actual Failure Frame but is individual for each model. These percentiles are obtained using the Pandas [55, 62] *dataframe.quantile()* function. After the percentiles and spreads are calculated, the interval is checked to ensure that the specific percentage of predictions is within the interval. The following states the intervals for 90%, 95%, and 99%. As before, the AlexNet 1_Img and 4_Img_1 models are not considered.

8.3.1 90%-Interval

The 90%-Interval metric uses the fifth and 95th percentile, as mentioned above. As the CNN models contain 179 testing experiments, at least 162, equating to 90.5%, must be within this interval to surpass the 90% margin. The results of the 90%-Interval are shown in Figure 32. The best result is achieved by the VGG16 4_Img_1 model with an interval spanning ten frames. ResNet50 with the same Data Model is within twelve frames and in combination with 4_Img_2 has 13 frames in the 90%-Interval. AlexNet shows the largest intervals and performs worst for each considered Data Model.

8.3.2 95%-Interval

This interval contains 95% of the predictions, for which the 2.5th and 97.5th percentiles are calculated to obtain the interval and number of frames described above. The min-

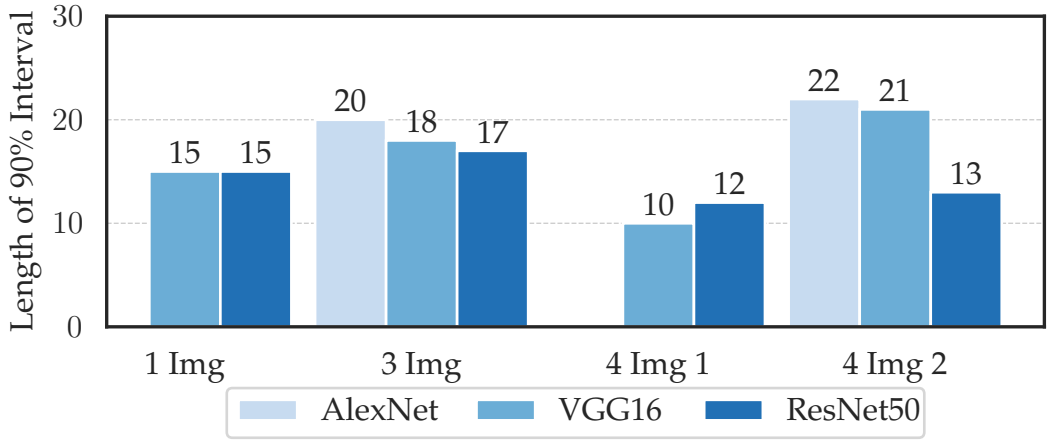


Figure 32: The 90%-Interval for the CNN models chosen are grouped by the Data Model and show the length of the interval, containing 90% of the predictions. Shorter intervals correspond to better performance.

imum number of experiments for this metric is 171, equating to 95.5%. The results, visualized in Figure 33, contain the number of frames within the 95%-Interval, continuing the trend seen in Figure 32. The combination of VGG16 with 4_Img_1 performs best with 14 frames in its interval, followed by ResNet50 with the same Data Model. However, when considering ResNet50 with 4_Img_2, the 95%-Interval contains 20 frames. From these results, Data Model 4_Img_1 significantly outperforms the other Data Models.

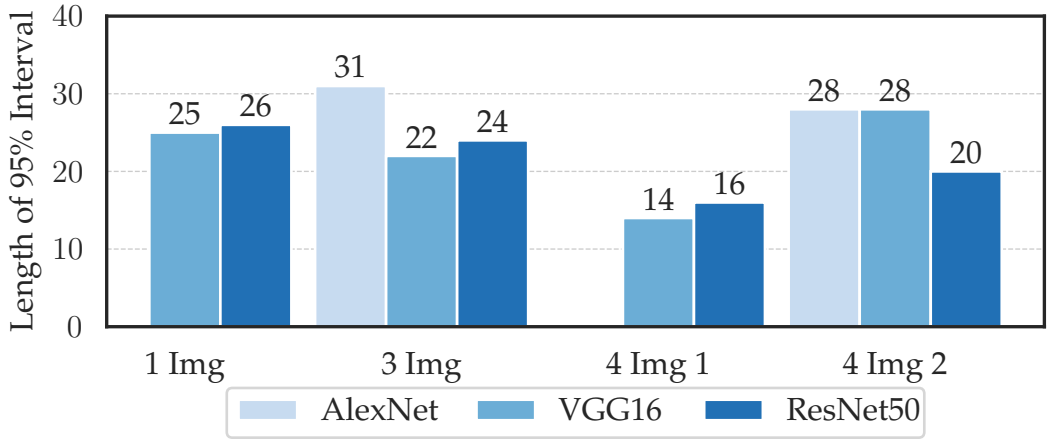


Figure 33: The 95%-Interval for the CNN models chosen are grouped by the Data Model and show the length of the interval, containing 95% of the predictions. Shorter intervals correspond to better performance.

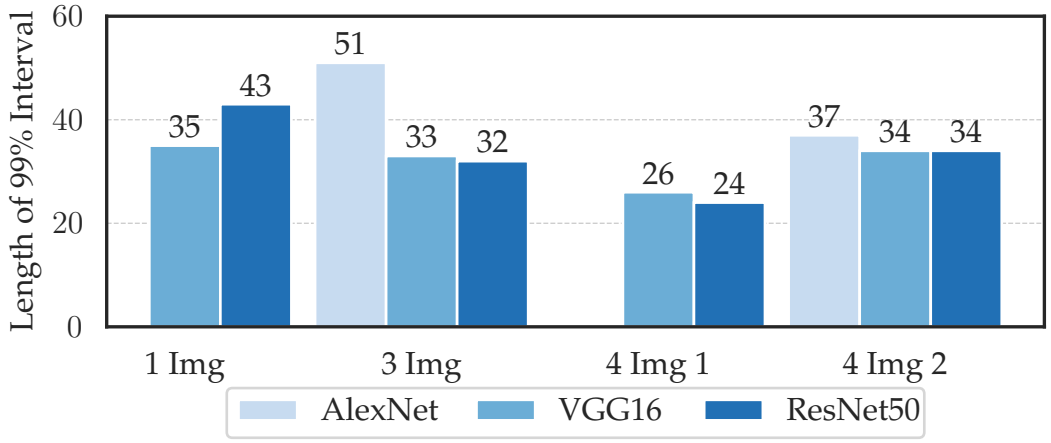


Figure 34: The 99%-Interval for the CNN models chosen are grouped by the Data Model and show the length of the interval, containing 99% of the predictions. Shorter intervals correspond to better performance.

8.3.3 99%-Interval

The 99%-Interval is naturally the widest and is calculated using the 0.5th and 99.5th percentile. 177 of 179 experiments, 98.88%, have to be within the interval, which is very close to the desired 99% and allows for two outliers. The resulting frame numbers within the intervals for the models are stated in Figure 34. While Data Model 4_Img_1 still achieves lower frame numbers than the other Data Models, the combination with ResNet50 has 24 frames in the interval. It outperforms the combination with VGG16, which has 26 frames in the 99%-Interval.

Concluding from the different intervals, Data Model 4_Img_1 achieves the smallest number of frames within the intervals compared to the other Data Models. The combination with VGG16 performs best at the frame numbers of the 90%- and 95%-Interval, paired with ResNet50, the 99%-Interval contains the fewest number of frames.

8.3.4 Worst Outliers

After presenting the predictions close to the Failure Frame, it is also essential to mention the worst predictions that have been treated as outliers in the metrics above, as in a possible application, these are relevant factors. The Worst Outliers are the two pre-

Table 4: The worst two outliers on both ends are stated as the difference to the Failure Frame for the considered CNN models.

	AlexNet		VGG16				ResNet50			
	3 Img	4 Img 2	1 Img	3 Img	4 Img 1	4 Img 2	1 Img	3 Img	4 Img 1	4 Img 2
min	-36	-23	-25	-18	-32	-21	-35	-20	-11	-43
max	19	29	28	24	16	23	22	21	16	25

dictions with the largest difference to the actual Failure Frame on both ends, located outside the 99%-Interval as shown in Table 4. VGG16 and ResNet50 with Data Model 4_Img_1 worst prediction is 16 frames after the Failure Frame, which is the best value for the outliers to the top. Whereas the model using ResNet50 also has the best value with eleven frames as an outlier on the lower side, the VGG16s outlier is at -32 frames. The Worst Outlier on the upper side is 29 frames coming from the AlexNet 4_Img_2 model, and on the lower side, -43 frames from the ResNet50 4_Img_2 model.

8.4 Inference Time

To evaluate a model, the time it takes to return a result is a significant factor. During inference, the time to assess one experiment is of interest for this use case, as there is only one experiment carried out and evaluated at once. The time for the three approaches is measured and compared.

The following procedure is applied to measure the time during inference. Four different experiments, each containing 100 images, are selected and evaluated ten times by each model. When considering the twelve CNN models, the simple image processing model, and the Haralick features model, a total of 560 tests are carried out. The following steps are timed: reading the data, generating the dataset, predicting and detecting changepoints, and writing the results to a .txt file in the experiments folder. Loading and compiling of a model is not timed, as this is only required once in an application case and does not scale with the number of evaluated experiments. The simple image

Table 5: Mean Inference Time for one experiment for the different approaches with the Apple M1 Pro and the server of the ILSB.

Model	Time Notebook (in s.)	Time Server ILSB (in s.)
Img_Processing	0.93	1.21
AlexNet	1.59	0.75
Haralick	3.10	9.93
ResNet50	4.74	2.65
VGG16	14.78	3.77

processing model uses no trained network; therefore, the processing time is measured. The recorded times are then averaged to obtain the mean prediction time.

Two different platforms are used to obtain the timing results: a notebook with an M1 Pro chip and 16GB of RAM and the student server at the ILSB with two Intel Xeon E5-2607 v3 14-core processors, 56 threads, and 792GB of RAM. The notebook is used because other processes and variables can be controlled better than on a shared server. After reviewing the results, the different Data Models in the CNN approach show no significant influence on the processing time and are therefore summarized for the used network architecture. The resulting times are listed in Table 5. The simple image processing model requires less than one second on the notebook and averages 1.21 seconds per experiment on the server. The Haralick model performs considerably well on the notebook but is the slowest model when running on the server at the ILSB. VGG16, however, is the slowest model on the notebook, with an average of 14.78 seconds per experiment, but is with an average time of 3.77 seconds, more than 3.9 times faster on the server. ResNet50 models require around 4.74 seconds on the notebook and 2.65 seconds on the server. A platform similar to the server will likely be chosen in a possible application. Therefore, all models, excluding the Haralick approach, require less than 4 seconds per experiment and could be implemented.

8.5 Showcase

A possible implementation is simulated in a showcase, and specific results are stated in the following. Unlike an actual implementation case, where the Failure Frame is obtained from the model, the previously labeled Failure Frame is used to evaluate the system's performance in the simulated use case. However, the principle can be translated to a usage without labels.

The experiment to be evaluated is chosen in the showcase file, and the approach is defined. As the CNN models show the best performance, only these models are used in the following. After selecting the experiment, the desired CNN and Data Models are defined. Then, the previously trained model is loaded, and the experiment data is pre-processed for processing by the model. The model then returns predictions for the dataset, which are then fed to the changepoint detection, where the Failure Frame is determined. In the optimal case, the obtained Failure Frame is accurate, marking a successful application. This successful case is shown in Figure 35, where the Failure Frame at frame 181 is correct. The used model for the prediction is VGG16 combined with the 4_Img_1 Data Model. In the visualization, a summary of the evaluated experiment is shown. At the top, the plot contains the label values of each frame symbolized with green dots, marking no failure with a zero value and a failure with a value of one. The blue line plots the predictions obtained from the classification process. The obtained values are discrete for each frame, but the values are connected with lines for better visualization. The green dashed vertical line marks the Failure Frame as labeled. The detected Failure Frame, as obtained from the changepoint detection, is represented by a cross that is either green or red, depending on the success of the prediction.

Four images are shown in the bottom section of Figure 35. These images can be used to verify the evaluation result visually. The second image contains the image of the predicted Failure Frame, with the image on the left showing the image one frame before. The image captured one step later is shown on the right side of the predicted

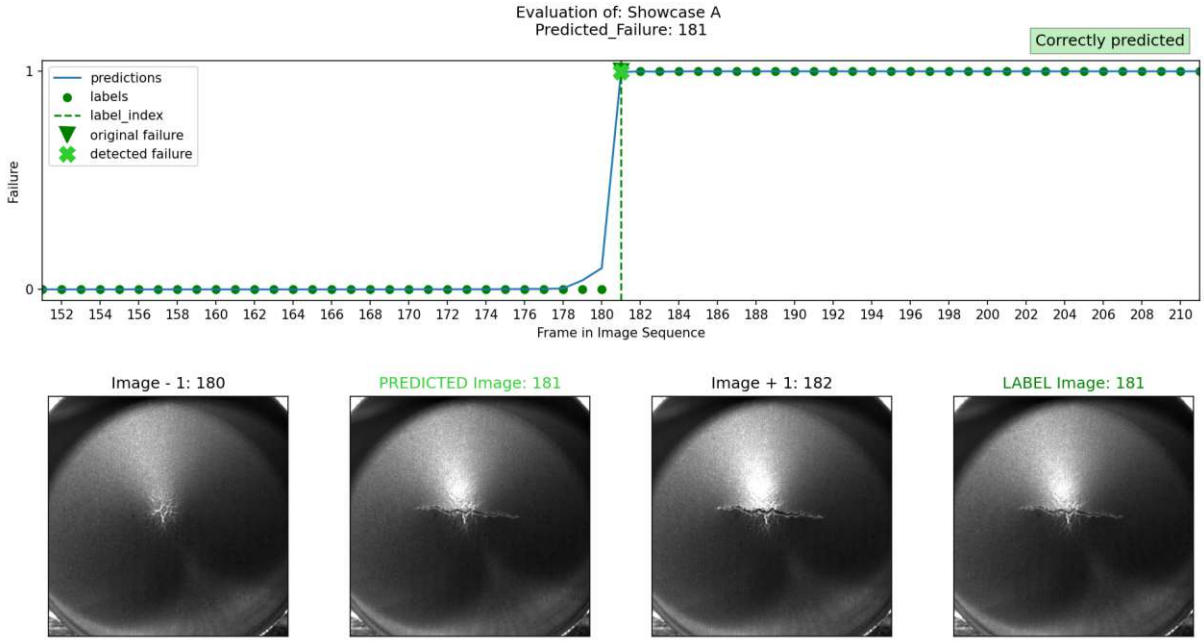


Figure 35: The evaluation of Showcase A is done with the VGG16 4_Img_1 model and predicts the Failure Frame correctly at frame 181. The predictions corresponding to the failure probabilities, as well as the labels, are shown. In the images at the bottom, specific images are shown to visually validate the result of the evaluation.

image. Therefore, these images can be used to detect changes in the images around the prediction result. With the labels available in this simulation, the right picture shows the labeled image.

In Figure 35, the classification predictions approximate the actual failure behavior relatively accurately, as the values are close to zero initially, and the change to values close to one is sudden and at the correct location in the series. Therefore, the changepoint can be detected correctly. When visually inspecting the failure behavior shown in the images at the bottom, the formation of a horizontal crack can be seen in the predicted image. In contrast, the image one frame previous shows the specimen in an intact state.

Showcase B also uses the VGG16 architecture, but the 1_Img Data Model and a different experiment is chosen. Figure 36 shows the evaluation result of this showcase. The predicted Failure Frame is also correct, as shown in the plot, and can be seen in the images at the bottom. However, when inspecting the classification predictions (blue

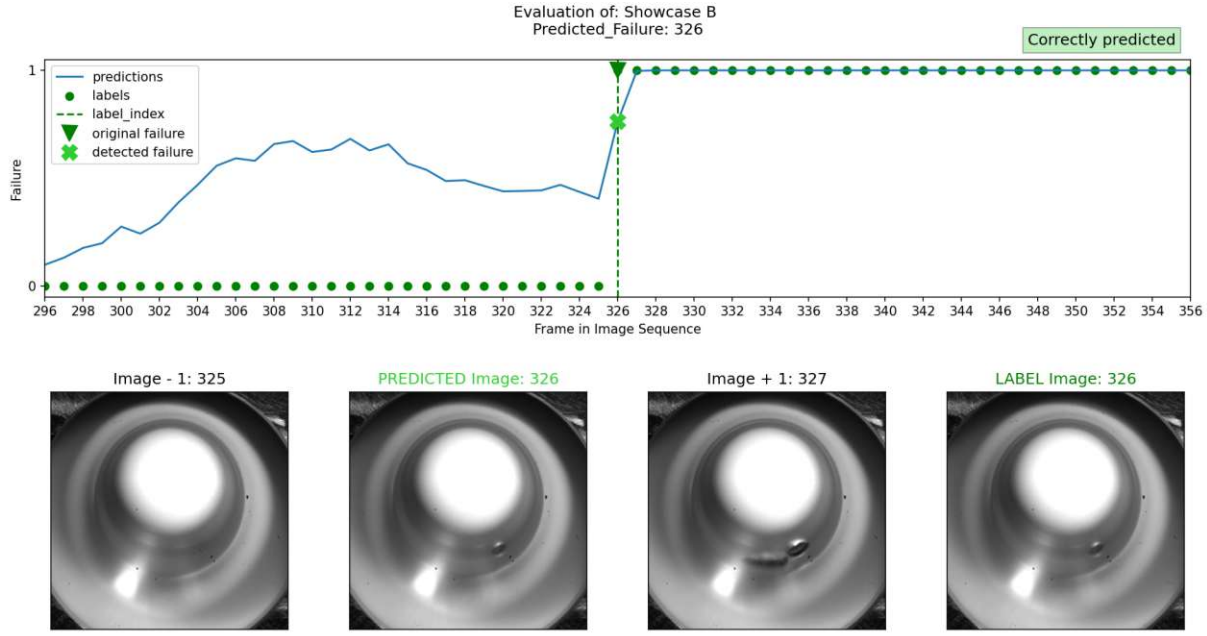


Figure 36: The evaluation of Showcase B is done with the VGG16 1_Img model and predicts the Failure Frame correctly at frame 326. The predictions corresponding to the failure probabilities, as well as the labels, are shown and are more inconsistent than in the evaluation of Showcase A. In the images at the bottom, specific images are shown to visually validate the result of the evaluation.

line) in the plot of Figure 36, determining the Failure Frame becomes more difficult. The changepoint detection is capable of identifying the correct location in the series of predictions, although the predictions are not close to zero before the Failure Frame. These high values for the classification predictions might originate from mirroring effects that occur as the reflective surface of the specimen bulges upon the impact and reflects light.

The Failure Frames of the two showcases above are predicted correctly, but as many incorrect predictions are also obtained, the worst predictions from the best two models are stated and discussed in the following.

Table 6: The Worst Predictions for the VGG16 and ResNet50 4_Img_1 models are listed in this table. The experiments, the difference between the predicted failure frame and the labeled failure frame, as well as the temperature of the specimen and insights from the inspection of these experiments are given.

Exp.	VGG16	ResNet50	Temp. [°C]	Inspection Insights
Sample A	0	-11	90	transparent, impactor visible
Sample B	12	2	-35	ice particles on surface
Sample C	16	8	23	reflective, ductile behavior
Sample D	5	16	80	very reflective, lighting position
Sample E	-32	0	80	very reflective, lighting position
Sample F	1	10	23	very reflective, lighting position
Sample G	-11	-8	23	many, extremely reflective areas
Sample H	15	8	23	reflective, lighting position
Sample I	-8	-11	-30	reflective, horizontal & vertical crack
Sample J	0	13	80	reflective, unclear Failure Frame
Sample K	-2	12	80	reflective, unclear Failure Frame
Sample L	12	12	23	very dark images

8.6 Worst Predictions for VGG16 and ResNet50 with Data Model 4_Img_1

According to the previous results, the best two models are the VGG16 4_Img_1 and the ResNet50 4_Img_1 model. However, when considering an implementation, analyzing the worst predictions a system returns is essential. Therefore, Table 6 lists all experiments, where either VGG16 or ResNet50 with Data Model 4_Img_1 predicts the Failure Frame incorrectly with a deviation of ten or more frames.

The Failure Frame from the first experiment (Sample A), as shown in Figure 37, is detected correctly by VGG16, but ResNet50 detects the Failure Frame eleven frames too early. Noticeable is the transparent specimen tested in this experiment, where the impactor can be seen before the specimen is hit.

The second experiment (Sample B) shows distinct ice particles on the surface of the specimen, which is very uncommon in the dataset. The Failure Frame of this experiment is detected two frames after the label by ResNet50 and twelve frames after the

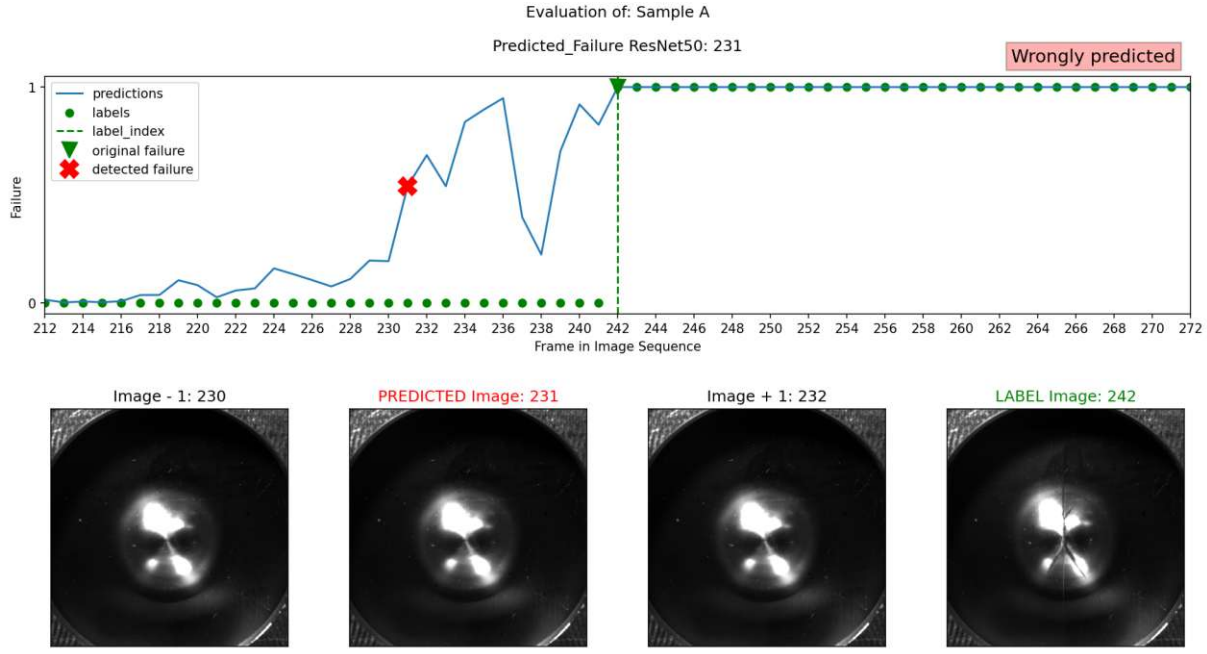


Figure 37: The transparent specimen in Sample A is mispredicted by ResNet50 4_Img_1 by eleven frames as marked with the red cross.

label by VGG16. The evaluation of this experiment (Sample B) with VGG16 is depicted in Figure 38.

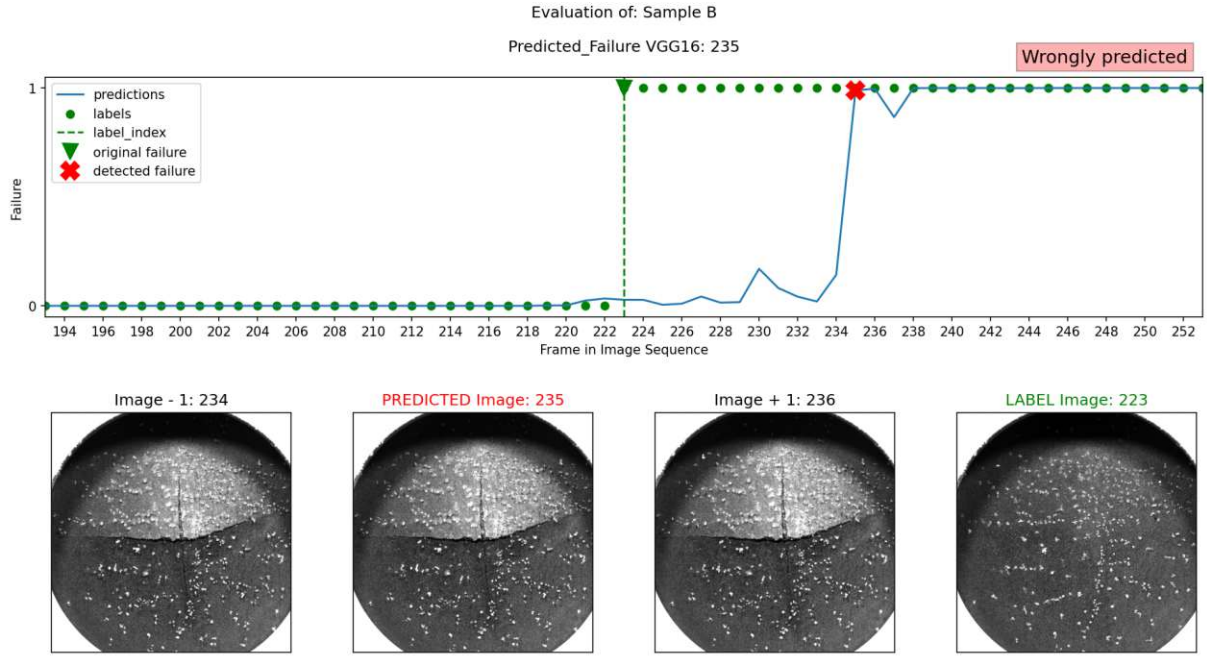


Figure 38: The formation of ice particles on Sample B may influence the prediction capabilities of the VGG16 4_Img_1 model.

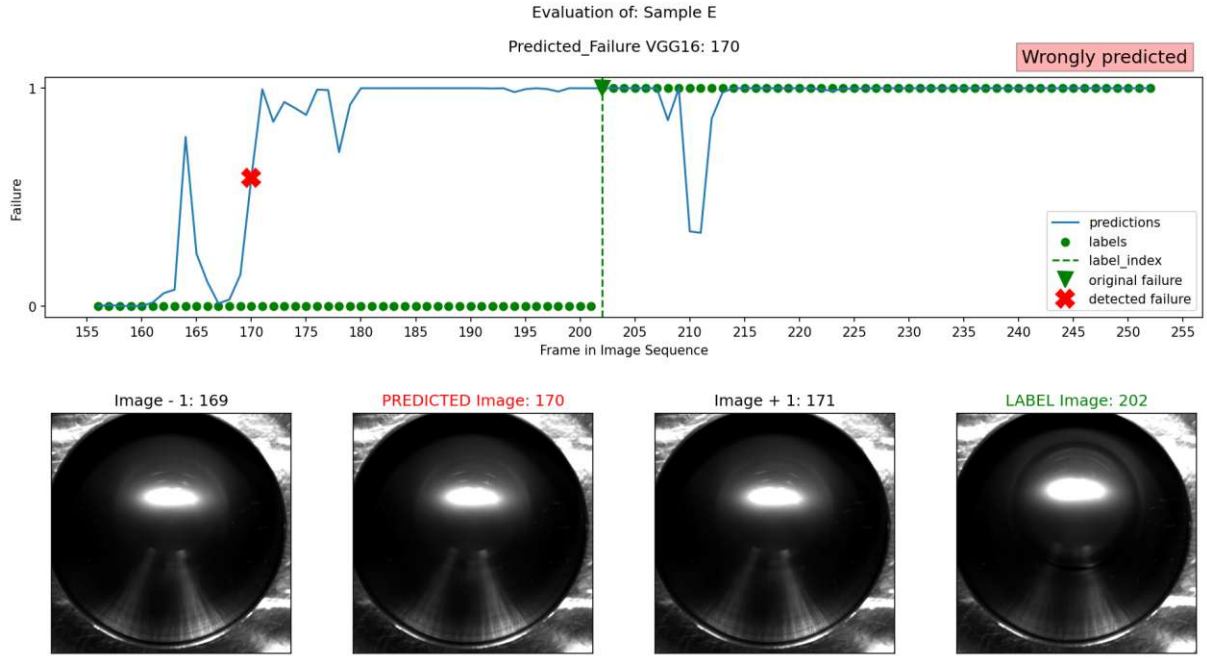


Figure 39: The specimen in Sample E shows reflective areas, which change over the image series. The prediction of the VGG16 4_Img_1 model is wrong by 32 frames. Additionally, the lighting is insufficient at the bottom of the bulge, where most ductile failures occur.

The following nine experiments all show reflective behavior, and most face lighting issues. As the lighting is directed slightly from above and the failure in ductile material often occurs below the bulge, this area is usually fairly dark. The evaluation of Sample E exhibits this behavior to an extreme extent, and the Failure Frame prediction for VGG16 is 32 frames off as shown in Figure 39, whereas ResNet50 detects the exact frame of the label.

Another noteworthy evaluation is found in Sample I, where the sample fails in two stages. The first phase is characterized by slowly progressing horizontal cracks, whereas a sudden vertical crack indicates the second phase. This second crack has been manually identified as the failure and chosen as the label Failure Frame. However, the first visible failure actually occurs in the first stage. VGG16 and ResNet50 both detect the failure in the first stage with the formation of more minor horizontal cracks. In both models, the classification predictions increase slightly at the labeled Failure Frame, as shown in Figure 40. However, the significant changes are within the first region.

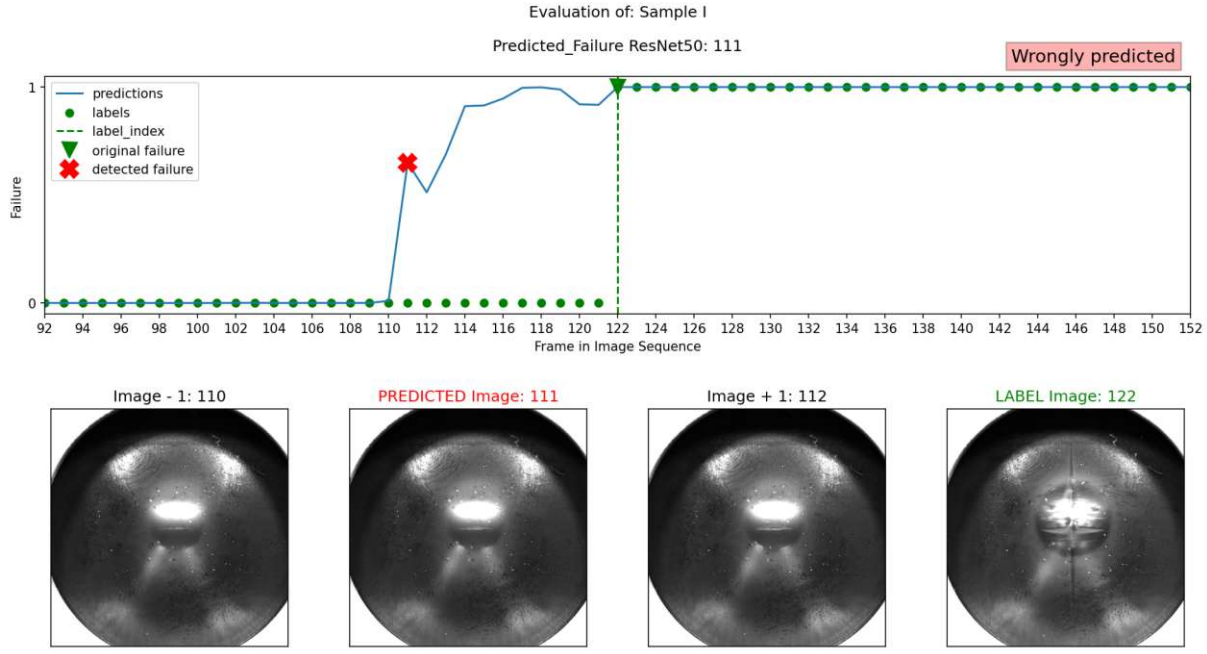


Figure 40: The specimen in Sample I forms cracks in two directions. While the horizontal direction evolves gradually and is detected by the ResNet50 4_Img_1 model, a second horizontal crack forms suddenly at a later point.

The assessment of Sample L yields an identical Failure Frame output from both VGG16 and ResNet50. However, the predicted failure occurs twelve frames after the designated Failure Frame. After reviewing the experiment images, it is evident that the lighting is inadequate, resulting in excessively dark images.

It is noticeable that most experiments that are predicted poorly by either VGG16 or ResNet50 when combined with Data Model 4_Img_1 show reflective behavior and lighting issues. Moreover, specific characteristics that are not very common in the dataset, such as ice particles and transparent specimen, seem to be difficult for an automated detection system.

9 Conclusion

This chapter summarizes and concludes this thesis. The information acquired during the Design Science research is communicated as follows, which completes the procedure described by the Design Science Research Methodology.

The main objective of this thesis is to develop an automatic system for Failure Frame detection from image frames of puncture tests which are captured by a high-speed camera. This system reduces the need for tedious and manual inspection of image series, resulting in more efficient and reliable evaluation of puncture tests. The development is carried out by implementing three different classification approaches, i.e. the simple image processing, the Haralick feature with Neural Network, and the Convolutional Neural Network approach. Each approach corresponds to a specific level of technology in Computer Vision approaches, ranging from traditionally engineered systems to state-of-the-art Deep Learning models. These different complexity levels in Computer Vision are compared to evaluate the beneficial effect of more complex models over simpler ones.

The developed systems consist of a classification on an image level and a subsequent segmentation of all the classification results from one experiment. Whereas the segmentation stage is always achieved using a changepoint detection, the classification is individual for the three approaches.

The selection of the classification approaches is supported by the insights from the literature analysis. The reviewed publications suggest the effectiveness of the AlexNet

and VGG16 architecture for the Convolutional Neural Network approach, along with the Transfer Learning concept.

The results indicate that models employing Convolutional Neural Networks show superior performance compared to the other approaches. The simple image processing approach is only able to capture some failure behavior, as its accuracy and reliability are not sufficient for practical use. However, it should be noted that this approach uses a very basic method to obtain the first-stage values, which are then processed by changepoint detection. Implementing a more advanced technique may enhance the accuracy of the Failure Frame predictions.

The second approach uses Haralick features and a Neural Network to determine the Failure Frame. This approach has posed a challenge during development because training the neural network on the Haralick features required cropping of the images to remove the specimen clamping. This increases the discriminative power of the Haralick features. Overall, this approach is not recommended for practical implementation due to the low performance exhibited by the results. Additionally, the computation of Haralick features demands considerable computational power.

Good performance values are achieved by Convolutional Neural Network models, especially the VGG16 4_Img_1 and ResNet50 4_Img_1 models, which perform best. These models are capable of predicting the Failure Frames correctly at the label frame in over 42% of the tested experiments. Most predictions are within a few frames around the labeled Failure Frame, with only a few distinct outliers. Over 88% of all predictions from these models are within five frames before or after the label Failure Frame, resulting in a maximum deviation of 500µs from the labeled Failure Time. Therefore, these two models are proposed for practical implementation. While the VGG16 model is more accurate close to the Failure Frame, the predictions of the model using ResNet50 are located within a narrower 99%-Interval.

While most Convolutional Neural Network models perform well, the AlexNet architecture in combination with Data Model 1_Img and 4_Img_1 fails in the training, as

these models are not adapting to the presented dataset. However, it has to be mentioned that all models are only trained once due to resource and time limitations, and far better results may be achieved by multiple rounds of training. Furthermore, no adjustments to the training parameters (hyperparameter tuning) were made due to the aforementioned time and resource constraints.

The data analysis reveals that the specimens within the given dataset demonstrate distinct failure behaviors, which leads to their categorization as Failure Cases. This categorization could facilitate the development of customized models targeting diverse failure behaviors. However, as the distribution of the Failure Cases is not even, and the dataset does not have enough experiments to train different models, this thesis has focused on developing one system for all experiments.

By evaluating experiments that are poorly predicted with VGG16 4_Img_1 or ResNet50 4_Img_1, difficulties such as lighting issues and reflections are obtained, which can be addressed to improve prediction performance. However, when considering the light reflection issues of many, especially ductile materials, an additional model for those experiments might be helpful.

Also, combining the results from the VGG16 4_Img_1 and ResNet50 4_Img_1 models could lead to more consistent and robust results. This is indicated as only one experiment is mispredicted by ten frames or more with both models. However, further research is needed to validate this hypothesis.

References

[1] 4a engineering GmbH. 4a test packages thermoplastic materials: setups & measurement definition v1.7, 2022. URL https://www.4a-engineering.at/downloads/testpackages.pdf.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Corrado, Greg, S.and Davis, Andy, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuanx Wicke, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/.

[3] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138(11):14–32, 2021. ISSN 08936080. doi: 10.1016/j.neunet.2021.01.026. URL http://arxiv.org/pdf/2005.00817v4.

[4] Hans-Jürgen Bargel, editor. *Werkstoffkunde: Strukturen - grundlegende Eigenschaften*. Springer-Lehrbuch. Springer Berlin Heidelberg, Berlin, Heidelberg, 13. aufl. 2022 edition, 2022. ISBN 978-3-662-63960-3.

[5] Ferdinand P. Beer, E. Russell Johnston, John T. DeWolf, and David F. Mazurek. *Mechanics of materials*. McGraw-Hill Education, New York, NY, eighth edition edition, 2020. ISBN 9781260113273.

[6] Alex Clark. Pillow (pil fork) documentation, 2015. URL https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[7] Luis Pedro Coelho. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software*, 1(1):e3, 2013. ISSN 2049-9647. doi: 10.5334/jors.ac.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

[9] Artur Fertschej, Peter Reithofer, and Michael Rollant. 4a impetus (PART 2): innovations – test methods, MAT_SAMP-1, anisotropy, composites and more. In *German LS-DYNA Forum 2016*, 2016. URL https://www.4a-engineering.at/downloads/events/28_rep_16110401_afer_pr_mr_gga_4aimpetusPart2_gesamt.pdf.

[10] Yuqing Gao and Khalid M. Mosalam. Deep transfer learning for image–based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768, 2018. ISSN 1093-9687. doi: 10.1111/mice.12363.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

[12] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 978-0262035613.

[13] Günter Gottstein. *Materialwissenschaft und Werkstofftechnik*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-36602-4. doi: 10.1007/978-3-642-36603-1.

[14] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018. ISSN 00313203. doi: 10.1016/j.patcog.2017.10.013.

[15] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6): 610–621, 1973. ISSN 0018-9472. doi: 10.1109/TSMC.1973.4309314.

[16] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández Del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. ISBN 1063-6919. doi: 10.1109/CVPR.2016.90.

[18] Alan R. Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[19] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75, 2004. ISSN 02767783. doi: 10.2307/25148625.

[20] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H.T.T. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 203–210. IEEE Comput. Soc, 2001. ISBN 0-7695-1119-8. doi: 10.1109/ICDM.2001.989520.

[21] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2020. doi: 10.1109/ACCESS.2019.2962617.

[22] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90020-8.

[23] Jie Huang, Qianting Hu, Zhenlong Song, Gongheng Zhang, Chao-Zhong Qin, Mingyang Wu, and Xiaodong Wang. Classification of cracking sources of different engineering media via machine learning. *Fatigue & Fracture of Engineering Materials & Structures*, 44(9):2475–2488, 2021. ISSN 1460-2695. doi: 10.1111/ffe.13528.

[24] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[25] Petr Hurtik, Stefania Tomasiello, Jan Hula, and David Hynar. Binary cross-entropy with dynamical clipping. *Neural Computing and Applications*, 34(14): 12029–12041, 2022. ISSN 0941-0643. doi: 10.1007/s00521-022-07091-x.

[26] Mahbub Hussain, Jordan J. Bird, and Diego R. Faria. A study on cnn transfer learning for image classification. In Ahmad Lotfi, Hamid Bouchachia, Alexander Gegov, Caroline Langensiepen, and Martin McGinnity, editors, *Advances in Com-*

putational Intelligence Systems, pages 191–202, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97982-3.

[27] ISO 6603-1:2000. Plastics — determination of puncture impact behaviour of rigid plastics: Part 1: Non-instrumented impact testing, 2000-03-09.

[28] ISO 6603-2:2023. Plastics — determination of puncture impact behaviour of rigid plastics: Part 2: Instrumented impact testing, 2023-06-02.

[29] Anil K. Jain. *Fundamentals of digital image processing*. Prentice-Hall information and system sciences series. Prentice-Hall, Englewood Cliffs, NJ [u.a.], 1989. ISBN 0133361659.

[30] Byunghyun Kim and Soojin Cho. Automated vision-based detection of cracks on concrete surfaces using a deep learning technique. *Sensors*, 18(10), 2018. doi: 10.3390/s18103452.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. URL http://arxiv.org/pdf/1412.6980v9.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc, 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[33] Tommy Löfstedt, Patrik Brynolfsson, Thomas Asklund, Tufve Nyholm, and Anders Garpebring. Gray-level invariant haralick texture features. *PloS one*, 14(2): e0212110, 2019. doi: 10.1371/journal.pone.0212110.

[34] Duo Ma, Hongyuan Fang, Niannian Wang, Binghan Xue, Jiaxiu Dong, and Fu Wang. A real-time crack detection algorithm for pavement based on cnn with

multiple feature layers. *Road Materials and Pavement Design*, 23(9):2115–2131, 2022. ISSN 1468-0629. doi: 10.1080/14680629.2021.1925578.

[35] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, 1995. ISSN 0167-9236. doi: 10.1016/0167-9236(94)00041-2. URL https://www.sciencedirect.com/science/article/pii/0167923694000412.

[36] Eizan Miyamoto and Thomas Merryman Jr. *Fast calculation of haralick texture features*. Technical report, Carnegie Mellon University, 2008.

[37] Adrien Müller, Nikos Karathanasopoulos, Christian C. Roth, and Dirk Mohr. Machine learning classifiers for surface crack detection in fracture experiments. *International Journal of Mechanical Sciences*, 209:106698, 2021. ISSN 0020-7403. doi: 10.1016/j.ijmecsci.2021.106698. URL https://www.sciencedirect.com/science/article/pii/S002074032100429X.

[38] Loris Nanni, Sheryl Brahnam, Stefano Ghidoni, Emanuele Menegatti, and Tonya Barrier. Different approaches for extracting information from the co-occurrence matrix. *PloS one*, 8(12):e83554, 2013. doi: 10.1371/journal.pone.0083554.

[39] Izaak Neutelings. Neural networks, 2021. URL https://tikz.net/neural_networks/.

[40] Nicoguaro. Typical stress vs. strain diagram for a ductile material (e.g. steel), 2020.

[41] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

[42] Raman B. Paranjape. Fundamental enhancement techniques. In Isaac N. Bankman, editor, *Handbook of Medical Imaging*, Biomedical Engineering Ser, pages 3–18. Elsevier Science & Technology, Burlington, 2000. ISBN 978-0-12-077790-7. doi: 10.1016/B978-012077790-7/50004-7.

[43] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007. ISSN 0742-1222. doi: 10.2753/MIS0742-1222240302.

[44] Lorenz M. Pfanner. *Project Work: Automatic evaluation of experiments based on machine learning*. unpublished project work, Institute of Lightweight Design and Structural Biomechanics, TU Wien, 2023.

[45] Richard Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image change detection algorithms: A systematic survey. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 14:294–307, 2005. doi: 10.1109/TIP.2004.838698.

[46] Rajagopalan-Sam Rajadurai and Su-Tae Kang. Automated vision-based crack detection on concrete surfaces using deep learning. *Applied Sciences*, 11(11), 2021. ISSN 2076-3417. doi: 10.3390/app11115229.

[47] Janosh Riebesell. Random tikz collection, 2022. URL https://github.com/janosh/tikz.

[48] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519.

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[50] Linda G. Shapiro and George C. Stockman. *Computer vision*. Prentice Hall, Upper Saddle River, NJ, 2001. ISBN 0130307963.

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. doi: 10.48550/ARXIV.1409.1556.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[53] David Stutz. Collection of latex resources and examples, 2022. URL https://github.com/davidstutz/latex-resources.

[54] Richard Szeliski. *Computer vision: Algorithms and applications*. Texts in computer science. Springer, Cham, second edition edition, 2022. ISBN 978-3030343712.

[55] The pandas development team. pandas-dev/pandas: Pandas, 2022. URL https://doi.org/10.5281/Zzenodo.7344967.

[56] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[57] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. ISSN 01651684. doi: 10.1016/j.sigpro.2019.107299.

[58] Usha Ruby, A., Theerthagiri, P., Jeena Jacob, I., Vamsidhar, Y. Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(4):5393–5397, 2020.

[59] Manisha Verma, Balasubramanian Raman, and Subrahmanyam Murala. Local extrema co-occurrence pattern for color and texture image retrieval. *Neurocomputing*, 165:255–269, 2015. doi: 10.1016/j.neucom.2015.03.015.

[60] Vidya Vijayan, Mareena Joy Chinsu, and Sivan Shailesh. A survey on surface crack detection in concretes using traditional, image processing, machine learning, and deep learning techniques. In *2021 International Conference on Communication, Control and Information Sciences (ICCISc)*, pages 1–6. IEEE, 2021. doi: 10.1109/ICCISc52257.2021.9484914.

[61] Michael Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021.

[62] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.